IMPROVED C3 LABORATORY CAPABILITIES FOR COMMAND AND CONTROL RESEARCH, ANALYSIS AND GAMING

Thomas Alan Secorsky
and
Thomas Patrick Stack

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMPROVED C3 LABORATORY CAPABILITIES FOR
COMMAND AND CONTROL RESEARCH,
ANALYSIS AND GAMING

by

Thomas Alan Secorsky
and
Thomas Patrick Stack

March 1981

Thesis Advisor: W. P. Hughes Jr.

Approved for public release, distribution unlimited

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): Improved C3 Laboratory Capabilities for Command and Control Research, Analysis and Gaming	5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis March 1981
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Thomas Alan Secorsky Thomas Patrick Stack	8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93940	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School Monterey, California 93940	12. REPORT DATE: March, 1981 13. NUMBER OF PAGES: 131
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

ACCAT	SIMULATION
WES	GAMING PHILOSOPHY
C3	ISL
C2 MODULE	APPANET
SEMINAR GAME	PDP 11/70

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis contains software, procedures and methodologies that can be utilized in the C3 Laboratory to experiment with and game command and control problems in the areas of organizational relationships, communications networks and analysis of procedures, combat doctrine or tactics. The demonstration game herein utilizes the Warfare Environment Simulator (WES), the organizational structure and concept of decentralized command embodied in the Composite Warfare Commander Doctrine, an automated scenario generator, software resident at ACCAT, and the primary software product of this thesis: a user-defined

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601 |

ck 20 (Cont'd) communications network supporting a multi-level chain of command
t allows for communications delays, garbled messages, message non-delivery,
player attrition.

Approved for public release, distribution unlimited

Improved C3 Laboratory Capabilities for Command and Control Research, Analysis and Gaming

by

Thomas Alan Secorsky
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1971

and

Thomas Patrick Stack
Lieutenant Colonel, United States Air Force
B.S., United States Air Force Academy, 1959

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY - C3

from the

NAVAL POSTGRADUATE SCHOOL
March 1981

ABSTRACT

This thesis contains software, procedures and methodologies that can be utilized in the C3 Laboratory to experiment with and game command and control problems in the areas of organizational relationships, communications networks and analysis of procedures, combat doctrine or tactics. The demonstration game herein utilizes the Warfare Environment Simulator (WES), the organizational structure and concept of decentralized command embodied in the Composite Warfare Commander Doctrine, an automated scenario generator, software resident at ACCAT, and the primary software product of this thesis: a user-defined communications network supporting a multi-level chain of command that allows for communications delays, garbled messages, message non-delivery, and player attrition.

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

aa	A Text Editor
ACCAT	Advanced C2 Architectural Testbed
ADM	Type of Terminal Device
Ann Arbor	Type of Terminal Device
ASM	Air to Surface Missile
AUTODIN	Automatic Digital Information Network
cr	Carriage Return
edn	A Text Editor
esc	Escape Key
IGL	Interactive Graphics Language
Genisco	Type of Graphics Display Device
MSG	A UNIX Subsystem
NOSC	Naval Ocean Systems Center
ORESTES	Cryptographic System
photint	Photographic Intelligence
RSM	Remote Site Module
server	Background Process used with ACCAT Graphics
sh	A Shell Command
SNDMSG	A UNIX Subsystem
SURVAV	Surface Ship Satellite Vulnerablility
Tektronics	Type of Terminal Device
TGO	Task Group ORESTES
XED	A Text Editor

ACKNOWLEDGEMENTS

The authors wish to express their gratitude to Mr. Phil Balma and LT Ellen Roland for their guidance and assistance during the preparation of this thesis; and to CAPT Wayne P. Hughes for the encouragement and understanding that made this thesis possible.

The authors would also like to acknowledge the assistance of our classmates who gave generously of their time and professional expertise in developing and demonstrating this thesis.

I. INTRODUCTION

The Secure Command, Control and Communications Exercise Laboratory was developed for use as a research, test and evaluation, and experimentation facility. Designated a Remote Site Module (RSM), the laboratory is part of a secure computer network involving the Naval Ocean Systems Center (NOSC), in San Diego, California, Commander-in-Chief, U.S. Pacific Fleet, in Pearl Harbor, Hawaii, and the Fleet Numerical Weather Facility, located in Monterey, California. The Naval Research Laboratory in Washington, D.C. is also slated to become a part of the secure network in the near future. The secure network is part of a larger network developed by the Defense Advanced Research Projects Agency known as the ARPANET. The Advanced Command and Control Architectural Testbed (ACCAT), located at NOSC provides a vehicle for conducting research, test and evaluation, and experimentation in state of the art hardware and software technologies.

The C3 Laboratory has the capability to utilize experimental software packages available at ACCAT such as SURVAV, LADDER, or QUERY3. The laboratory is also regularly utilized in demonstrations of CINCPACFLT's Warfare Environment Simulator (WES), as well as other wargames and simulations, in an effort to identify those areas in which wargaming or simulation, utilizing

advanced computer hardware and software, can highlight command and control relationships, processes and problem areas.

The primary focus of this thesis is the demonstration of software, procedures, and methodologies that can be utilized in the C3 Laboratory to experiment with and game command and control problems in the areas of organizational relationships, communications networks, and analysis of procedures, combat doctrine, or tactics. The demonstration game utilizes the Warfare Environment Simulator (WES), the organizational structure and concept of decentralized control embodied in the Composite Warfare Commander Doctrine, an automated scenario generator, software resident at ACCAT, and the primary software product of this thesis, a user-defined communications network supporting a multilevel chain of command that allows for communications delays, garbled messages, message non-delivery, and player attrition. It has been assumed that potential readers have some familiarity with an interactive computer operating system.

II. SEMINAR GAMING

One type of wargame that can be played without the use of sophisticated batch process simulations or interactive computer wargames is the seminar game. A seminar game is simply a time sequenced, structured scenario. The scenario is implemented by messages, normally handed to the addressee at the proper game time by the game controller. The messages must be constructed so as to generate actions and reactions among the participants that are consistent with the purpose and intent of the game. The messages that unfold the scenario are usually in the form of intelligence summaries, or broad directives from "higher authority."

The seminar game is particularly useful for command and control research and analysis. The structure and implementation of the scenario can be designed to provide data for postgame analysis of information processing requirements, decision-making criteria and methodology, and organizational relationships. The major deficiency in a seminar game with only manual processes is the inability to efficiently and effectively monitor the interactions between players or groups of players for later analysis.

Utilization of the C3 laboratory facilities for seminar gaming can eliminate the necessity for hand delivering messages and provide the means to obtain a transcript of the

interactions of all the players, both formal and informal. The game controller can play an active or passive role or both. By playing the role of "higher authority" he can be in a position to provide realistic direction to the game and steer game play to fulfill scenario objectives. Seminar gaming in the C3 laboratory allows for a wide range of scenarios specifically designed to provide data for analysis of command and control areas of interest, as well as supporting functions such as communications and computer support.

Automation of the seminar game can be accomplished by exploitation of the UNIX operating system resident on the laboratory PDP 11/70 computer, the TOPS20 and TENEX operating systems on the NOSC computers, or both. Inherent in the UNIX operating system are system commands, referred to as macros, which execute subroutines within the operating system itself. Of particular value is the fact that these macros are simple, easy to understand and execute commands which require no knowledge of any programming language. The more sophisticated operating systems on the NOSC computers have several useful subsystems which also can be directly applied to the gaming environment.

In constructing a seminar game it is necessary to understand the physical limitations of the laboratory. There are twenty-two locations hardwired to the the computer, limiting computer monitoring of game interactions to twenty-two terminals. Figure 1 depicts the lab configuration. The letters on each

desk are the characters by which the computer knows each desk connection. Directories can readily be established by the laboratory computer staff to support any scenario. These directories allow any number of players up to twenty-two to log onto the system using their game name (i.e. JCS, CINCPAC, CINCSAC), and be able to receive messages addressed to them under that game name. Additional hard copy terminals can and should be brought into the laboratory to augment the video display terminals in residence, and in particular to provide hard copy capability at selected stations.

There are several ways of constructing a seminar game. The game can be message driven; that is, the stimuli for game interaction are provided by predetermined messages designed to unfold the scenario to the players in whatever manner the game designer desires. The advantage of this rigid scripting is that it allows repeated identical scenarios for controlled tests. In a controller driven game, the scenario is outlined for the players, along with initial conditions, and an active exchange between the players and the game controller can provide the stimuli for player interaction. An advantage of this type of game is that the game controller, acting as "higher authority", has more flexibility in driving the scenario towards the game objectives, and the players benefit from enhanced realism in the game environment. A third method of implementing a seminar game is one of game play against a thinking opponent, typified by tactical trainers with BLUE

versus ORANGE scenarios. The latter two methods of playing a seminar game can provide the ability to superimpose a seminar game on a more rigidly structured gaming support system such as WES. The integration of a computerized wargame with a seminar game has the potential for providing a more realistic environment for command and control research and analysis.

III. MOTIVATION FOR C2 GAMING

The importance of command and control is highlighted by the findings of a colloquium sponsored by the Secretary of Defense in 1979 which recognized the need to develop a conceptual framework for analyzing, designing, and evaluating military command and control systems, and assessing their capability in combat environments. In short, it was agreed: that due to the complexity of the military organizaiton and command structure there was no generally applicable systems theory; that a command and control system cannot be separated from the human it supports, and a better understanding of the human decision making process could affect technology requirements; that it is important to distinguish between wartime and peacetime command and control requirements; and finally, that measuring the performance of a command and control system in a combat environment and evaluating the contribution of command and control to overall force effectiveness, are both areas requiring attention. Singled out as being of particular importance is our inability to do comparative analysis of U.S. and Soviet command and control. The treatment of command and control as a separate warfare area by the Soviet Union indicates they expect to gain a potential advantage in capability. [Ref. 1]

Three areas for research that the participants of the colloquium felt could be of significant value in attempting to redress these shortcomings were suggested. The first involves obtaining a better understanding of the decision-maker, his requirements and weaknesses, and the man-machine interface. The second involves experimentation with various organization structures and command relationships. The third was the development of methodology to evaluate command and control in a combat environment, particularly in a counter-C3 environment. [Ref. 1]

The inability to define what is meant by command and control, or by command, control and communications, or by command, control, communications, and computers, or still yet command, control, communications, and intelligence, on a national level, has seriously hampered efforts to assess the relative worth of command and control programs and systems. The confusion generated within the command and control community adds fuel to the fires of competitors for defense dollars, and strengthens the convictions of those who are not proponents of enhancing our capabilities. The temptation to view C3 as nothing but a conglomeration of sophisticated communications systems, computer networks, and sundry other electronic devices whose sole purpose is to allow higher echelons of command to usurp traditional command decision-making authority any time anything out of

the ordinary occurs, is strong. One who obviously fell into this trap has stated:

> "What does C2 and C3 and now C3I mean in the combat environment of tomorrow? It means that we have a marvelous electronic suger teat that our CO's, OTC's and CINC's are being trained to suck on and from which there isn't the slightest possibility of their being weaned - until, of course, it's too late. We'll wean ourselves away after we have tangled with an adversary powerful and clever enough to break our electronic crutches at just the right moment, isolate us tactically as independent units, and then force us in our crippled state into combat on terms dictated by him." [Ref. 3]

Viewing command and control as merely electronics is a misperception. The degree of sophistication of electonics systems properly derives from our goals of acquiring, processing, and understanding information faster and "better", in order to gain an advantage over an opponent. Graceful degradation of system performance is one of the key design goals of any system designed to support command and control. We have always had command and control in some form or other. The supporting systems have evolved over the years as the state of our technology has progressed. Command and control includes also, many supporting functions which contribute every bit as much as electronics. The mobility of force components, state of training and the way we train, development and testing of doctrine and tactics, operational procedures, and organizational and command structures and relationships are just a few of the "non-electronic" components of command and control which deserve consideration

in the evaluation of the effectiveness of command and control and its contribution to overall force effectiveness. [Ref. 4]

IV. C2 MODELS AND GAMING PHILOSOPHY

A. DEFINITION

In order to design a scenario for a seminar game that will allow for command and control related analysis, a definition of the game designer's perception of what command and control means is of primary importance. The definition of command and control used in designing the demonstration scenario in this thesis is as follows:

> COMMAND - The planning and decision process and the resultant actions in the form of a plan and orders, direction or guidance for its implementation.
>
> CONTROL - The attempt to make the plan come true under conditions of stability and orderliness. [Ref. 5]

B. C2 MODELS

Models of command and control systems have several interesting and identifiable features. Some of these are:

1. Hierarchy

Command Control, especially in a military sense, is hierarchical in nature. The relationships among the levels of that hierarchy are of great interest when we are interested in examining the effectiveness of a decision-making process.

2. Process

Command Control can be viewed as a process through which information is provided to a decision-maker, decisions are made and are implemented. An examination of this process is required. In addition, an examination of the date from which information is derived is of great interest.

3. Mission/Environment

From an operational viewpoint, models of military Command Control systems are bounded by the stated military mission. What is that mission? Is it to fight General War, Threater Nuclear War or General Nuclear War? Is it to fight or provide appropriate military options to the highest government levels during periods of crisis? Would the same system be used to support all these missions, and if not, what capabilities should the systems have for transition from one mission state to another? How long should each element of the system be able to survive?

From a cybernetic point of view, a Command Control model envisions information coming from the operational environment and being processed or filtered in some manner before it reaches the decision-maker. The decision (Command), once made, is in turn processed and an attempt at decision implementation (Control) is made. The decision changes the environment and the cycle continues. Overlaid upon the information gathering activity is the factor of uncertainty regarding the accuracy of the information gained from the data collected. The

decision-maker is not in direct contact with the full filtered or distorted information.

Models exist in order to provide a basis for evaluation of the completeness of an existing system to meet the needs of the user. Models exist to direct the effort toward the design of new systems. Models are useful as aids in making comparisons and contrasts among existing or proposed systems. Will the systems function in the postulated environment? Will the system support the operational mission? Will it help the commander to make "better" decisions?

C. GAMING OBJECTIVES

A primary objective of the seminar game in this thesis is the simulation of command and control decision processes. Restricting the players to communicating only by electronic mail has several advantages in analyzing decision-making processes. A decision-maker that has no method of informally communicating with a supporting staff is forced to request information from either a database or one of the other players. The ability to recover this information at the conclusion of the game negates the need for large numbers of observers attempting to monitor the informal communications of all of the decision-makers playing the game, as well as presenting the information requirements of the decision-maker. Additionally, insight can be gained into how the decision-maker perceives the situation by the questions

he asks. The interactions between different organizations and hierarchies, and their respective methodologies, is exposed for analysis. The ability to stress players at different levels in the command heirarchy by shortening the period allowed for the planning and decision functions allows for analysis of decision-making under pressure in a state of uncertainty.

A second primary objective of this seminar game is to demonstrate some of the methodology and techniques of employing the assets of the C3 laboratory in C2 gaming. The game designer has great flexibility in his ability to model communications networks and schemes, as well as both horizontal and vertical echelons of command. Command and control processes can be simulated at the lowest level of command to the highest, and any mix in between.

A third primary objective is to stimulate interest in the use of C3 laboratory techniques for command and control research and gaming. It has long been recognized by experts in the field of wargaming that the analytic models used to represent decision-making, out of necessity, assume that the decision-maker is logical, has perfect information, and acts on a predictable schedule. This is obviously not the case. A seminar game, designed specifically to provide information on the requirements of command and control, superimposed on a computer gaming support system such as

WES, is one method available to address the issues highlighted by the colloquium as requiring further research and experimentation.

D. THE DEMONSTRATION SCENARIO

The operations order with annexes contained in Appendix A was developed for the purpose of providing a framework for a seminar game to demonstrate the capabilities developed by this thesis project. [Ref. 6] The scenario was designed to include as many of the available methodologies and technologies as possible in a reasonably believable manner. It is expected that certain of the actions in the scenario will cause player responses which will provide insight into the command and control process. A confused transition from an exercise state to one of alert, and the necessity to redefine roles and missions, rules of engagement and organizational relationships is expected. The interactions of the warfare commanders operating under the Composite Warfare Commander procedures is expected to provide some insight into the coordination required to conduct disparate missions simultaneously with decentralized control. The transition to a hot war environment, and the ensuing period of freeplay is expected to demonstrate the feasibility of gaming tactical command and control problems in a warfare environment utilizing WES as an environment simulator and analytic tool, rather than as a tactical trainer.

V. SYSTEMS COMMUNICATIONS CAPABILITIES

There are several ways to send a message on the Secure Network. The foremost of these is the message handling system inherent in the operating system, called MSG. The MSG sublevel of the operating system keeps all messages for a particular account in a file "MAILBOX", and allows the account holder to send a message to anyone else with an account by the SNDMSG routine. Another method is by linking active terminals together by the LINK (TENEX operating system) or TALK (TOPS2Ø operating system) commands. This feature allows any number of players to interconnect their terminals, with each capable of talking in real time to all of the other players. One disadvantage of this method of communication is that the players must utilize the computer systems at NOSC. A third method, similar to LINK and TALK but resident locally in the UNIX operating system is the WRITE macro. The WRITE command allows one player to send a message to any other player that is currently logged on the system. A disadvantage of this method of communication is that there is no simple way to retrieve these messages at the completion of game play unless they were between players on hard copy terminals, or the TELNET TYPESCRIPT feature of NOSC's computer facilities is utilized. Selection of a schema for communicating is dependent on the design of the game

scenario, availability of the computer systems at NOSC, or the desire to use in-house facilities.

A simple demonstration of the use of the WRITE command and the UNIX operating system to act as a message generator for a seminar game can be viewed by logging into the system as NCA (National Command Authorities) on one terminal, CPACFLT on another, and CLANFLT on yet another. These directories, as well as the directories listed in Appendix A, have been established by the laboratory computer staff in support of this thesis. They all share the same password and several other characteristics: a mailbox and a file called ".PROFILE" which sets the terminal display to twenty-four lines (the maximum field of view on the smaller video terminals), announces the presence of new mail upon log-in, and lists other users currently logged in by terminal location. Each directory also has a file named "LOOKMAIL." This file is a compiled C language program that runs as a background process for ten hours after log-in. The program provides notification to a player when a message has arrived in his mailbox. On the terminal logged in as NCA enter the following commands after the UNIX prompt, the percent sign (%):

sh<tryout (carriage return).

The command "sh<tryout" invokes the UNIX operating system SHELL. It is in fact the SHELL which responds to any command following the UNIX prompt. In this case the SHELL is

specifically told to execute the commands contained in the file "tryout." The file "tryout" contains the following commands:

```
find -name msgdemo1;
sh<msgdemo1;
sleep 2Ø;
find -name msgdemo2;
sh<msgdemo2;
sleep 2Ø;
find -name msgdemo3;
sh<msgdemo3;
```

The operating system has simply been told to find a file called "msgdemo1" and execute any commands contained therein, wait 2Ø seconds and repeat the sequence for the file "msgdemo2" and so on. The "msgdemo" files invoke the WRITE command followed by the address of the player you wish to send the message to (in this case the player's game name), and a plain text message in any format the designer desires. File "msgdemo3" is a demonstration of a broadcast message to everyone on the network. Contents of this file or any other of the demonstration files can be viewed by calling the files into any editing system. This then, is a relatively simple method of storing and sequentially executing messages to unfold a scenario in a seminar game. The simplicity of the WRITE command and relative ease in which files can be executed by the SHELL is the simplest method of

scenario execution. A basic seminar game can be designed utilizing files executable by the SHELL to generate the scenario or provide the stimuli for player interaction. The modularity of the laboratory allows the game designer to simulate various echelons of command, organization of staff functions, and communications schema. Use of the WRITE command between players can simulate any number of communications methods. The WRITE command could, for instance, represent as AUTOSEVOCOM network, a simplex Task Group Orestes (TGO) secure HF/UHF teletype network, or simple Autovon or commercial telephone network. The major disadvantage of this type of message generator is that the message is not retrievable by a player not on a hard copy terminal.

The TOPS 2Ø operating system at NOSC allows for the use of RUNFIL and MSG as a message generator. Use of MSG is advantageous in that it allows players not on hard copy terminals to review any message received at any time. A RUNFIL is a command file executed by the operating system. A demonstration of a message generator using this methodology can be viewed by logging onto the system as NCA, CPACFLT, and CLANFLT as before. As NCA execute the following sequence of instructions:

(1) TELNET TOPS2Ø;

(2) LOG esc NPS3 esc PASSWORD esc esc (carriage return);

(3) Type RUNFIL, and when asked for file name type TEST.RUNFIL (carriage return).

The file "TEST.RUNFIL" contains commands to enter MSG and respond to all system queries just as if a user were manually using MSG. The delay between execution of responses is to allow time for system response to the commands being entered. A disadvantage of this method of generating messages is that system response time varies and is not predictable, and enough time must be given to each response to ensure the proper sequencing of commands. The file "TEST.RUNFIL" was constructed in the TOPS 2Ø text editor, XED, and contains the following:

```
*MSG
*^^^2ØØØØ
*S
*^^^5ØØØ
*CPACFLT@NPS
*^^^5ØØØ
*CLANFLT@NPS
*^^^5ØØØ
*TEST OF TOPS 2Ø MESSAGE GENERATOR
*^^^5ØØØ
*^BF^^2ØØØ (control b, F, control up-carat, up-carat,
            up-carat)
*DEMO1
*^^^2ØØØØ
*
*^Z (control v, control z)
```

```
*^^^5ØØØ
*
*^^^2ØØØØ
*S (REPEAT SEQUENCE FOR DEMO2)
```

The commands must exactly follow the responses required by the operating system for manual use of MSG. In addition, the time delays entered, in milliseconds, must be sufficient to allow the system to execute the previous command before accepting another command. To enter the time delay requires a special sequence of commands:

(1) Control up-carat (no response should be visible on the line);

(2) Up-carat (produces ^);

(3) Up-carat, (produces ^^);

(4) Enter delay in milliseconds.

Control z is the command to exit XED as well as the command to send a message. A control v followed by control z tells XED to disregard the control z as an XED command, allowing it to be interpreted as a command by MSG.

VI. A DETAILED C2 MODULE

A. PURPOSE

The message handler is an organizational and communications module which exists as a software program residing on the PDP 11/70 computer in the C3 laboratory at the Naval Postgraduate School. The goal of the module is to provide a vehicle through which an experimenter may conduct inquiry into some of the interesting features of Command Control models. The message handler is modular in that it exists independently of, and will operate independently of, other modules. It is possible that other independent modules such as situational displays and data base retrieval schemes will be developed which will interact with this module to provide an even better research vehicle.

The module exists as a universal Command Control "front-end" to an operational environment. The model will demonstrate its utility as the filter between the various commanders in the Caribbean scenario (Appendix A) and the operational environment which is represented by the Warfare Environment Simulator. WES was selected, as discussed earlier, for several reasons. Among them: WES exists and is operable in the C3 laboratory; WES has applicability for those who wish to conduct inquiries into Naval Tactics. Heretofore WES has been used at the Postgraduate School mainly for demonstration since it has such a vivid graphical display capability.

The Caribbean scenario could easily be replaced by an open ocean Carrier Battle Group defense against ASM scenario where the message handler would model the intra- and inter-fleet communications and the WES data base would be configured to look at the positioning and sensitivity of fleet over-the-horizon sensor assets. Conversely, the module could as easily function in "front" of a Theater Air-Land combat environment or in a scenario-driven high echelon planning game.

B. PROGRAM DESCRIPTION AND CAPABILITIES

The program consists of three main parts:

1. A game builder portion called "WARGAME" controlled by an umpire in which organizational and communications parameters are set and can be changed by the umpire.

2. A game play portion called "PLAY" in which appropriate players communicate with each other within the organizational and communications structure set by the umpire.

3. The message transmission program called "MAILER". MAILER is an asynchronous program which executes silently as a background process at each player position.

The overall architecture of the module (Fig. 2) allows the "building" and "modifying" portions of WARGAME to write model parameters into various files. The "playing" portion, which is resident at each player's position, reads these parameters and operates on them in the manner described below. The "transmission" portion continuously reads the delivery times of messages written in PLAY, compares the delivery times

to the current clock time and forwards those messages whose time has elapsed. The actual message transmission is accomplished by a scaled-down version of UNIX SNDMSG called TRANS. TRANS also operates silently and forwards only those messages whose parameters have been read by MAILER. One of the interesting technical questions imbedded in the execution of this module is how well the PDP 11/70 will be able to handle a multi-station wargame of this nature. The answer to this question, although only a byproduct of this endeavor, should serve to help answer technical questions which will arise when the overall architecture of the C3 laboratory is evaluated by some energetic and competent student doing thesis research at some future date.

WARGAME is the experimenter's personal program. It has several parts:

1. In the portion called "Build" he can specify: (a) The organizational structure he wants to exist in the experiment; (b) The communications links which exist between players in that structure; and (c) The "quality" of those links. The "quality" of the link is entered for each player pair as a number zero through six. The value zero means that there is no direct link between that player pair. The value six means there is "perfect" (uninterrupted and undegraded) communications between the pair. The values one through five are assigned to the following representative types of communications links. It is important for the reader to understand that the links one through five only represent the types of communications circuits listed. There are no separate queuing algorithms for each individual link. All utilize the same single server algorithm wich is discussed just following subparagraph 4 below. The

experimenter controls the queing which actually takes place by the values he assigns to the average message arrival rates and average message service rates which are appropriate at each communications node.

(1) Encrypted Landline with precedence control.

(2) Non-Encrypted Landline with precedence control.

(3) Digital RF (HF/UHF/VHF) with A/J resistance.

(4) Digital RF (HF/UHF/VHF) without A/J resistance.

(5) Analog Voice.

(6) Perfect Link (No Delay, No Degradation).

2. Also available in "Build" are two subprograms which utilize the Genisco color graphics display devices. The first, called "Floorplan" simply displays the C3 laboratory floorplan showing the current locations of the Ann Arbor, ADM and Tektronics I/O devices. The second, called "Diagram 1" allows the experimenter to graphically build, change and display a three-tiered organizational structure. There is no interaction between the build actions in Diagram 1 and the organizational construction described in paragraph above. The purpose of the diagram subroutine is simply visual display.

3. In the portion called " Modify" the experimenter can specify the average message arrival rate at a given facility and the average message service rate at that facility. It is unlikely that the amount of traffic generated during game play will cause the queuing of messages. Certainly whatever queue there is will not be reproducable. The message arrival rates and service rates introduced by the experimenter are surrogates for the varying traffic densities in an actual communications system. How these rates are used by the module is explained below. The experimenter can also specify the rate at which messages are to be "lost" over a given link and the rate at which messages are to be "garbled" in transmission over that link. He may also specify the names of players to be removed from the game because they have either been "destroyed" or because their communications

facility is no longer functioning. In addition to establishing these message arrival rates, the experimenter may change any of these paramenters during the course of the game.

4. The umpire can also send messages to players. These messages will not be affected by the queuing algorithms contained in PLAY.

The message arrival rates and service rates are used by the program to calculate a message queuing time. The queuing time represents the amount of time a message will be delayed in arriving at its destination and is calculated from the single-server queuing theory equation:

$$wq = a/s(s-a)$$

where wq is the average queuing time; a is the average message arrival rate; and s is the average message arrival rate. Unscheduled arrival rates for messages are viewed as conforming to a Poisson distribution and the inter-arrival times between these messages as following a negative exponential distribution. [Ref. 7] During game play the program transforms a uniformly distributed random number (y) to an exponentially distributed random number (x) using the relationship:

$$x = (-1/L)\ \ln(y)$$

where L is the average message arrival rate specified by the experimenter. [Ref. 8]

Messages to be lost or garbled in transmission are expressed by the experimenter as a percentage of the total number of messages transmitted from one communications node to any other. That is, if the experimenter wants three percent of the messages over a given link class to be lost, he enters the number three. During play, the program calculates a uniformly distributed random number each time a message is sent from one player to another. If that random number is equal to or less than the specified rate the message is "lost" or "garbled." For garbled messages only the text is affected. If a message is garbled, the scrambled text enters the queuing calculation and is sent. If the message is to be lost it is not sent to the addressee. The sender does not see either of these calculations.

Players may be removed from the game by the umpire during game play. The listing of players to be removed is additive. That is, if player "A" had been previously removed and the umpire later wished to remove player "B", he would enter "Modify" and type only player "B's" name. The file containing the list of removed players would be added to, not overwritten.

PLAY is the portion of the module belonging to the individual players. Each player logs in at his playing position giving his player name as the login name and the player password "turtle." Play offers the player a choice of reading messages which have been sent to him or of transmitting messages to other players in the game.

If the player chooses to read his mail the program will fork to the UNIX subsystem MSG. MSG will first provide a summary list of message headers and then, if requested, access the text of any message using well-known MSG commands. The player should use MSG only to review messages written to him. Sending messages from this portion of the program, although possible, is not acceptable since it will bypass all the message queuing algorithms which are written into the program.

When the player chooses to communicate with other players, a message header automatically appears and the player fills it out at the keyboard. The program makes a comparison of the identity of the sender and the identity of the principal addressee and searches a two-dimensional array for the value of the communications link between them. If the value of that link is zero, the sender is advised that no direct communications path exists and that his message should be routed through the organization to which he belongs. This attribute of the program offers the experimenter a powerful tool, especially in a multi-service environment, for investigation of the operational effects which may result from the inability of one commander to communicate directly (and speedily) with another. If the value of the link is six, the program bypasses all the queuing calculations and sends the message directly to the addressee. This attribute of the program will allow the investigator to examine organizational issues independently of communications uncertainties.

If the value of the link is read as one through five, the program enters the segment in which queuing time is calculated and in which the decision is made whether the message is to be lost or garbled in the communications system.

Queuing time is calculated from the single-server queuing theory equation previously stated. The arrival rate for unscheduled arrivals follows a Poisson distribution and the inter-arrival time (y) follows a negative exponential distribution using the following relationships:

$$f(y) = Ae^{-Ay}, \quad E[y] = 1/A \text{ and } Var[y] = 1/A$$

where A is the mean of the message arrival rate. [Ref. 7] The service time is also negative exponentially distributed with a mean of 1/S where S is the average service rate. We assume an infinite population of messages. It is necessary that the ratio A/S be less than 1 or the queue and the waiting times will increase without bound.

The program converts an uniformly distributed random number to an exponentially distributed number and calculates a sample arrival rate and a sample service rate based upon the mean rates supplied by the umpire. These sample rates produce a queuing time. The queuing time is added to the computer-generated clock time to form the time the message should arrive at its destination. This arrival time, called "sendtime", along with the name of the action addressee, the names of the information addressees, the subject and the

text of the message, are written into one of five queuing files which are located in the player's directory. The information in the queued file waits to be recognized by the program MAILER and sent to its destination.

If the message is to be lost, it is not sent to the action addressee, but to a central file of lost messages called "lostfile" which is located in the umpire's directory. If the message is to be garbled in transmission, the text of the message is randomly overwritten by the character "&" before it is queued for transmission. Lost messages take precedence over garbled messages. That is, if, randomly, a message meets both the lost message rate parameter and the garbled message parameter, it will enter the lost message portion of the program and the garbled message portion of the program will be bypassed. None of these calculations is seen by the sender.

MAILER is a program which executes asynchronously with PLAY in each player's directory. [Ref. 9] The program continuously rereads the sendtimes which are written as the first data item in each queuing file. When MAILER finds a sendtime which is equal to or greater than the current computer clock time, it reopens the queuing file and reads the action addressee, the information addressees and the subject. It also reads the file containing the text of the message. These character strings are carried in a fork to a greatly modified version of UNIX SNDMSG called TRANS [Ref. 10], which also

resides in the umpire's directory. TRANS accepts only the data items which are carried in fork from MAILER and does not announce its presence to the system.

Data Collection: In addition to the summary file of lost messages in the umpire's directory, the umpire is a silent information addressee on every message sent from any player to any other player. The message sender is also a silent addressee on all messages he sends. Further, each time a message is sent anywhere in the system a summary of the message parameters is written into a file called "data" which is also located in the umpire's directory. The information written into "data" is as follows:

1. From
2. To
3. Subject
4. Sendtime (clocktime plus queuing time)
5. Queuing Time
6. Message Disposition
    a. Sent
    b. Lost
    c. Garbled
    d. Queued (resubmitted)

The "data" file is not designed to be read by the umpire using an editor such as "aa" or "edn" since in the interest of code simplicity, all end-of-string characters have been

left attached to the strings which are written into the file. The end-of-string characters do not appear, however, if the command "cat data" is issued at the keyboard or the contents of the "data" file are directed to a printer.

It is possible, then, for the experimenter by looking at: (1) The mailfile of each individual player; (2) The summary files of lost messages (lostfile); and (3) The summary file of message parameters (data), to reconstruct the exact flow of message traffic from player to player. The umpire's own composite mailfile serves as a backup to the individual player's mailfiles. By comparing the flow of messages and the game scenario, the experimenter has the opportunity to evaluate the hypothesis under which he originally established the organizational and communications system parameters.

C. USER INSTRUCTIONS

1. Compilers

All the programs which exist within the Command Control model are written in the "C" language. [Ref. 11] Not all of these programs are compiled using the same compiler command. Wargame.c, Play.c and Mailer.c are compiled using "cc". Trans.c is compiled using the compiler "cct":

```
pcc -n $1 dateparse.o /sys/source
      /mailsys/libJ/libj -ly
```

In addition, the following must be in the directory when cct is used:

msgdefs.h send.h dateparse.o

The subprograms Floorplan.c and Diagraml.c use the compiler "kcc":

```
cc $1.c /usr/graphics/genlib.a/usr
/graphics/genlib.a/usr/graphics/genlib.a
```

This compiler allows more streamlined graphics commands to be used in the Genisco source code.

2. Directories

Files should be located in the directories as listed:

control:

Wargame/wargame.c	loss1
trans/trans.c/trans.o	loss2
send.h	garb1
dateparse.o	garb2
msgdefs.h	data
msgrate	lostfile
lossrate	destroyed
garbrate	mailbox

cntrlf:

diagraml/diagraml.c	index
floorplan/floorplan.c	matrix
pick1	pick1a
pick2	pick2a
pick3	pick3a
pick4	pick4a
pick5	pick5a
prerate	number

Each player's directory:

play	.profile⟨mailer&⟩
mailer	mailbox
header1	text1
header2	text2
header3	text3
header4	text4
header5	text5

3. Using Build and Modify in WARGAME

This segment of the paper will examine the game building and modifying instructions contained in WARGAME.

a. The Main Menu

The first choices presented to the experimenter are in the main menu. BUILD allows initialization of player names, organization and communications links. MODIFY permits initialization of, and changes (during game play) to message parameters. The experimenter may exit the program at any menu level by typing (q)uit.

WHAT PORTION OF WARGAME DO YOU WISH TO ACCESS?

BUILD (B)
MODIFY (M)
PLAY (P)

b. The Internal Password

The player is required to enter an internal password at this point to gain access to either the BUILD or the MODIFY portions of the program. The wargame password

is "tstack." This paragraph is the only place this password is listed in the open. The contents of the password were removed from the source code after the program was compiled for the last time.

PLEASE ENTER THE WARGAME PASSWORD

c. The BUILD Menu

The selections are as indicated. The programs containing the graphics for the lab floorplan and the organizational structure are accessed via a fork. The selection "DESIGN THE COMM NET ...." is discussed in detail in sub paragraphs d and e.

THIS PORTION OF THE PROGRAM WILL ALLOW THE UMPIRE TO:

1. REVIEW THE LAB FLOORPLAN (F)
2. DESIGN THE ORGANIZATIONAL STRUCTURE FOR THE GAME (O)
3. DESIGN THE COMM NET SUPPORTING THE ORGANIZATION (C)

d. Identifying the Players

The program asks for the number of players. There is an arbitrary limit of 20 players set currently. This number is a function of the number of terminal locations available in the lab. The program will echo the value entered. If the experimenter enters a number larger than 20, the program will not accept the value and will issue an advisory.

HOW MANY PLAYERS, INCLUDING THE UMPIRE, ARE THERE?
THE NUMBER OF PLAYERS IS:
ONLY 20 PLAYING POSITIONS PER SESSION!

The program asks for the players' names and echoes the responses. The inputs must be in lower case since the transmission system ultimately uses a modified version of UNIX SNDMSG which recognizes participant's names in lower case only.

WHAT ARE THE NAMES OF THE PLAYERS IN THE GAME?
(no caps)
PLAYER NUMBER [ ] IS:
YOU HAVE NAMED [ ] PLAYERS

e. Establishing the Values for the Communications Links

The program requires a numerical value to express the "quality" of the communications link between each player pair. The first selection is to establish a "perfect" link for all players by typing "all". If this choice is not appropriate, type "cr" and individual choices by pair will be offered.

IF YOU WISH ALL CIRCUITS TO HAVE THE SAME VALUE AND ALL PLAYERS TO BE ABLE TO COMMUNICATE WITH ALL OTHER PLAYERS DIRECTLY, TYPE "all" OTHERWISE, TYPE "cr".

For pair-by-pair value, the program automatically sets the quality of each link to Ø (no link). The program then searches the name list of players and offers a pair-by-pair entry for link values. If the experimenter desires to leave the value

as Ø, he depresses "cr". Otherwise, he enters the value from from the table below. For all entries, the program echoes the selected value to the operator. If the operator enters any value other than Ø-6, the program issues an advisory and reoffers him the selection.

ENTER THE CIRCUIT QUALITY FOR EACH PAIR
OF PLAYERS AS SHOWN IN THE LEGEND BELOW:

Ø -- None (No Communications Link)
1 -- Encrypted Landline with Precedence Control
2 -- Non-Encrypted Landline (AUTODIN)
3 -- Digital RF Circuits (HF/VHF/UHF) with A/J
4 -- Digital RF Circuits without A/J protection
5 -- Voice
6 -- Perfect Link (No Delay, No Degradation)

ALL COMM LINKS HAVE BEEN PRESET TO Ø. SELECT THE VALUE REQUIRED AS THE CHOICE IS PRESENTED TO YOU PLUS "cr" (If Ø is the correct value, type "cr" only).

THE CIRCUIT VALUE BETWEEN [ ] AND [ ] IS:
THE VALUE [ ] IS NOT ACCEPTABLE
THE VALUE IS NOW:

f. Returning to the Main Menu

This is the end of the BUILD section of WARGAME. Return to the main menu by depressing "cr".

RETURN TO THE MAIN MENU BY DEPRESSING "cr"

g. The MODIFY segment

MODIFY offers 5 parameters for initialization or for change during play.

PARAMETER INITIALIZATION AND MODIFICATION SUBROUTINE

THIS PORTION OF THE PROGRAM ALLOWS THE UMPIRE TO:

1. Choose which message arrival and message service rates he wishes to use in the game.
2. Establish initial values for these rates.
3. Establish the frequency for lost messages.
4. Establish the frequency for garbled messages.
5. Remove players from the game.

BASED ON THE QUEUING ALGORITHM FOR SINGLE SERVER FACILITIES THE AVERAGE AMOUNT OF TRANSMISSION DELAY FOR MESSAGES ADDRESSED TO A GIVEN FACILITY (WQ) CAN BE EXPRESSED AS A FUNCTION OF THE AVERAGE MESSAGE ARRIVAL RATE (A) AND THE AVERAGE MESSAGE SERVICE RATE (S).

$$WQ = A/S(S-A)$$

h. Average Arrival and Service Rates for Messages

The experimenter has two basic choices for message rates. He may choose to enter the actual rates for each circuit type or he may choose general rates. The general rates option is designed to give the experimenter a method of approximation if he wishes to bound his experiment before entering specific data. If he wishes to bypass the Arrival Rate segment he may type "cr" and go on to the next segment.

YOU MAY SPECIFY THE ACTUAL ARRIVAL RATES AND SERVICE RATES FOR EACH CLASS OF COMMUNICATIONS (1 through 5) OR YOU MAY RELY UPON A PRE-ESTABLISHED SERVICE RATE AND ARRIVAL RATE RELATIONSHIP AND VARY ONLY THE MESSAGE ARRIVAL RATE BY REQUESTING:

a. Normal Traffic

b. Medium Traffic (twice the normal arrival rate)

c. Heavy Traffic (three times the normal rate)

TYPE "1" TO INSERT SPECIFIC ARRIVAL AND SERVICE RATES

TYPE "2" TO USE THE GENERAL RATES (Normal, Medium, Heavy)

NOTE: IF YOU DO NOT WISH TO CHANGE VALUES CURRENTLY SET -- TYPE "cr"

i. Specific Arrival and Service Rates

TO AVOID A QUEUE WHICH GROWS WITHOUT BOUND, INSURE $A/S < 1$

NOTE: MESSAGE RATES ARE IN NUMBERS OF MESSAGES/ MINUTE. USE REAL NUMBERS OF 99.99 OR LESS.

Specific Rates:

FOR CIRCUIT QUALITY [ ]:

A =

S =

j. Standard Rates

The numbers used in calculation of queuing times if "standard" rates are selected are shown in the table below. The experimenter has the option to modify these standard rates by a factor of 2 or a factor of 3. If "normal" arrival

rate is selected the calculations take place as shown below. If "medium" traffic is selected, the arrival rate is doubled with respect to listed relationships. If "heavy" traffic is selected, the arrival rate is tripled.

> THE PRE-ESTABLISHED RELATIONSHIP BETWEEN ARRIVAL AND SERVICE RATES FOR A NORMAL ARRIVAL RATE IS AS FOLLOWS:
>
> For circuit quality 1: S = 3.10A
>
> For circuit quality 2: S = 3.05A
>
> For circuit quality 3: S = 3.04A
>
> For circuit quality 4: S = 3.03A
>
> For circuit quality 5: S = 3.02A
>
> ENTER "NORMAL", "MEDIUM" OR "HEAVY" TO ESTABLISH THE INITIAL MESSAGE ARRIVAL RATE.
>
> CHANGES TO THESE VALUES WHICH ARE DESIRED DURING THE GAME SHOULD BE MADE BY RE-ENTERING "MODIFY" OR BY EDITING THE FILE NAMED "PRERATE" IN THE DIRECTORY NAMED "cntrlf" USING THE PASSWORD "wargame".

k. Moving to the Next Segment

> TO CONTINUE, DEPRESS "cr"

(1) Lost Message Rates. The experimenter enters the rate at which he wants messages to be "lost" during transmission. He may enter a seperate loss rate for each circuit class (1-5) or he may enter a single rate which applies equally to all classes of circuit. If the values already entered are satisfactory, he may skip past this segment by typing "cr".

ENTER THE RATE FOR MESSAGES TO BE "LOST." FIVE EQUALS FIVE PERCENT. USE INTEGER VALUES.

TYPE "1" IF YOU WISH TO ENTER A SEPARATE LOSS RATE FOR EACH CIRCUIT TYPE. (1-5)

TYPE "2" IF YOU WANT A STANDARD RATE FOR ALL CIRCUITS

NOTE: IF YOU DO NOT WISH TO CHANGE VALUES CURRENTLY SET -- TYPE "cr"

ENTER THE LOSS RATES BY COMM CIRCUIT QUALITY FOR CIRCUIT QUALITY [ ]:

LOSSRATE =

or:

ENTER THE STANDARD RATE FOR ALL CIRCUITS

LOSSRATE =

CHANGES TO THESE VALUES WHICH ARE DESIRED DURING THE GAME SHOULD BE MADE BY RE-ENTERING "MODIFY" OR BY EDITING THE FILE NAMED "lossrate" IN THE DIRECTORY NAMED "control" USING THE PASSWORD "wargame".

m. Garbled Message Rates

The experimenter enters the rate at which he wants messages to be "garbled" during transmission. He may enter a separate rate for each circuit class (1-5) or he may enter a single rate which applies equally to all classes of circuit. If the values already entered are satisfactory, he may skip past this segment by typing "cr".

ENTER THE RATE AT WHICH YOU WISH MESSAGES TO BE "GARBLED" DURING TRANSMISSION. FIVE EQUALS FIVE PERCENT. USE INTEGER VALUES.

TYPE "1" IF YOU WISH TO ENTER A SEPARATE RATE FOR EACH CIRCUIT TYPE. (1-5)

TYPE "2" IF YOU WANT A STANDARD RATE FOR ALL CIRCUITS.

NOTE: IF YOU DO NOT WISH TO CHANGE THE VALUES CURRENTLY SET, TYPE "cr"

ENTER THE MESSAGE GARBLED RATES FOR EACH CIRCUIT

FOR CIRCUIT QUALITY [ ]:

RATE =

or:

ENTER THE STANDARD RATE FOR ALL CIRCUITS

RATE =

CHANGES TO THESE VALUES WHICH ARE DESIRED DURING THE GAME SHOULD BE MADE BY RE-ENTERING "Modify" OR BY EDITING THE FILE "garbrate" IN THE DIRECTORY NAMED "control" USING THE PASSWORD "wargame".

n. Removing Players From the Game

Players may be removed from the game at will. Enter the number of players to be removed, then their names as the program asks for them. Names entered will be echoed by the program. If no player removal is desired, "cr" will pass this segment by.

ENTER THE NUMBER OF PLAYERS AND THE NAME OF EACH PLAYER WHO HAS BEEN "DESTROYED" OR WHO FOR SOME REASON IS TO BE REMOVED FROM THE GAME AFTER GAME START. ENTER THE NUMBER AND A "cr" AND THE NAME OF EACH PLAYER WITH A "cr" FOLLOWING EACH NAME.

NOTE: IF YOU DO NOT WISH TO CHANGE THE VALUES CURRENTLY SET -- TYPE "cr"

NUMBER OF PLAYER(S) TO BE REMOVED =

PLAYER NAME:

REMOVED PLAYER IS:

o. Return to the Main Menu

This is the end of the MODIFY portion of WARGAME. Return to the main menu by depressing "cr".

RETURN TO THE MAIN MENU BY DEPRESSING "cr"

4. Using Send and Read in PLAY

This portion of the paper will examine the commands in the message reading and message sending portions of PLAY.

a. The Menu

The choices offered to the player are to send a message, read messages which have been sent to him or of exiting the program. The program will accept either upper or lower case responses. There is a default advisory if any selection other than "s", "r" or "q" is made.

WELCOME TO WARGAME!

TO SEND A MESSAGE TYPE "s"

TO READ YOUR MAIL TYPE "r"

TO EXIT THE PROGRAM TYPE "q"

b. Reading Messages

In order to permit the player to read his mailfile without continually exiting the program and entering "mailbox" and then reentering the program to send messages, the program will fork to the subsystem MSG. MSG will provide the player first, with a list of the headers of all messages which have been sent to him; and second, allow him to read those messages by executing standard APPANET MSG commands. It, of course, is possible to send messages to other players while in MSG. This is not appropriate however, since all message queuing and degrade calculations will be bypassed. MSG is implemented only for reading. Furthermore, the player should not delete any of the messages he has read, since each player's individual mailfile is an important product in the post-game analysis.

YOU WILL FIRST SEE A LISTING OF THE HEADERS OF MESSAGES IN YOUR MAILBOX.

1. TO SEE THE TEXT OF ANY GIVEN MESSAGE, TYPE "t" FOLLOWED BY "esc" AND THE MESSAGE NUMBER. (Same actions as in the "MSG" System)

2. AFTER READING MESSAGES TYPE "q" TO RETURN.

3. DO NOT ATTEMPT TO SEND MESSAGES FROM THIS PART OF THE GAME.

c. Sending Messages

The more intricate portion of the program involves the sending of messages in PLAY. As has been discussed in detail in this chapter, many message corruption calculations

take place here. All are invisible to the player. However, in some cases the results of the calculations produce an advisory to the player. For example, in the message header the player inserts his name and the action addressee's name in lower case. A comparison of these names is made to the established communications quality matrix to find the value to the link between the players. If the link value is zero, the player is advised that no direct link exists so that he can reroute his message. The program also makes a comparison here with the listing of players who have been "destroyed" and if there is a match, an advisory is issued that the circuit between them is no longer in service.

MESSAGE HEADER (no caps)

FROM?:

TO?:

THERE IS NO DIRECT CIRCUIT TO [ ]

Route the message through your organization

COMMUNICATION WITH [ ] HAS BEEN LOST.

The remainder of the message header is filled out using the prompting commands listed below. Note that a comma is required after the name of each information addressee, even if there is only one. This is because there are two silent information addressees in the information string. Commas are required to separately identify each for transmission by by TRANS. The precedence prompt only appears if the circuit

between sender and receiver is a precedence control circuit. The subject line can be 50 characters in length. Those beyond 50 are truncated.

INFO: (comma after EVERY entry)

SUBJECT:

PRECEDENCE:

Following the message header, the player is prompted to insert the text of his message. He indicates termination of text with the "*" symbol. The symbol was chosen because of the low likelihood of its use in text. After the text string is finished, the player may review his message and choose either to send it or to erase it. As is discussed in Section D of this chapter, if all the queuing files are busy when the player elects to send his message, the BUSY advisory is presented. Otherwise, the message is sent -- subject to the degrade calculations operating in the background.

MESSAGE TEXT: (Type "*" When Finished)

SEND OR ABORT?

ALL THE COMMUNICATIONS CIRCUITS ARE BUSY. PLEASE WAIT A FEW MINUTES AND RESUBMIT YOUR MESSAGE.

c. Returning to the Menu

Finally, at the end of the message sending activity, the player is offered the opportunity to return to the menu and start the entire process once again.

TO CONTINUE, DEPRESS "cr"

D. POSSIBLE IMPROVEMENTS

Several areas in the program can be improved. Some of the improvements could take the form of refinements to the code currently in existence. Others could take the form of additions to the code -- additions which will offer the experimenter other tools with which to work.

The technique used for storing messages in files waiting to be recognized by MAILER is functional although there are limitations which a more skillful programmer could overcome without difficulty. Presently, there are five files at each player's position in which messages wait for their calculated delivery time to elapse. Sample queuing times from tests range from a fraction of a minute to better than 40 minutes as a function of the rates and the circuit values specified by the experimenter. It is possible that at any given moment, all the waiting files at a given player's position could be full. If this happens, the program presently advises the player that his communications "facility" cannot accept any more messages and requests him to wait a few minutes and resubmit his message. This type of "front-end" delay is arbitrary and not predictable. It may inject a confusion factor in later reconstruction of events by the experimenter. The architecture for improving this facet of the program could be to write the queued messages into a single "limitless" file. This would require keeping track of each message seperately by

counting total bytes written. The mailer program would then be required to seek and successfully identify the boundaries between the queued messages before forwarding them.

It may be valuable to provide the experimenter with the option to have message traffic "intercepted" by players to whom the traffic is not addressed. This capability is not specifically a portion of the two general classes of inquiry which were stated as being the purpose of the module, e.g.: (1) Inquiry into organizational issues as they may affect decision-making and, (2) Inquiry into the effects of delayed or degraded communications on decision-making. However, this command and control module has little utility on its own. It is designed to function as the filter to some environment and as such, the experimenter may desire to have the option to give information to the "opposition" to serve as the catalyst to interaction of forces. The framework for silent addressing already exists in PLAY. That is, copies of all messages are currently sent to both the sender and to the umpire's mailboxes without any action required by the player. This additional addressing is fixed in the program. Implementation of a variable silent addressing scheme will require changes in both WARGAME and PLAY. WARGAME could be modified to offer the umpire the capability to write the names of players who are to "intercept" messages into a file. This file must be selectively accessible by all players via PLAY. That is the umpire must also be able to specify by name the players whose messages

are to be intercepted. Attention must be paid here to the number of read/write files allowable per directory. That limit seems to be fourteen files per directory. The directory "cntrlf" is currently full and the directory "control" contains ten read/write files.

In PLAY the player is currently asked to provide the precedence of his message if that message is to travel over a link whose quality is designated as quality one or quality two (Encrypted/Non-Encrypted Digital Landline). The precedence is a dummy value and has no effect on the delivery paramenters for a message. Since the program currently computes a queuing time based upon a transform of a uniformly distributed random number, it is probable that the queuing times for consecutive messages from a given sender will vary. If precedence control is to be placed upon these messages queued for delivery, two basic changes must be made. PLAY must be changed to write the message precedence (if applicable) into the queuing files along with the sendtime, the addressee name, the information addressee names and the subject. MAILER must be changed to read not only the sendtime but the precedence and then search other queuing files for messages of higher precedence from that sender. If it finds such messages it must make them available to TRANS even though their sendtime will not have arrived.

VII. GRAPHICS SUPPORT

CHART and TEXT, resident on the TOPS2Ø system at ACCAT, is an on-line mapping program. Mapping coverage is world-wide with user defined latitude, longitude and scale, and on-line instructions. To access CHART and TEXT log on to the Tektronixs 4Ø14 and type the following instructions:

(1) Telnet to TOPS2Ø;

(2) Login NPS3;

(3) Type CHART;

(4) Type TEK in response to the query for a device parameter file.

The device parameter file "TEK" identifies the user as being on a Tektronix 4Ø14 terminal at NPS UNIX and the symbology necessary to establish the computer-to-computer path. The on-line instructions lead the user through construction of the chart. After saving the chart desired the program returns the user to the executive level. To add text and symbology to the chart type TEXT. TEXT provides on-line instructions to read in the desired chart and save it when completed. Since the saved charts are automatically put into common user files it is necessary to rename the file of the chart. To rename the chart type:

(1) RENAME (old fild name)(new file name .GPX)

The new file name must end in ".GPX" to identify it as a graphics file. To view graphics files produced by CHART and TEXT on the GENISCO screens a sequence of special commands must be entered at NPS prior to accessing ACCAT. To demonstrate those commands and view a graphics file, type the following:

(1) /usr/wes/startup (Clears all screens)

(2) sh<maprunfilØ (Starts special graphics processes and identifies GENISCO Ø as the port)

(3) telnet to TOPS2Ø

(4) log in to NPS3

(5) type RUNFIL

(6) type CCF2.GPX

The RUNFIL "CCF2.GPX" accesses a graphics language subsystem called ⟨LEVEL2⟩VIEWER, specifies the device parameter file, and calls in the graphics file to be viewed. This RUNFIL uses the same control commands as the message generator RUNFIL. Use of the RUNFIL technique allows the user to build a package of charts which can be automatically sequenced in time or called up on demand, with the RUNFIL performing the task of accessing the subsystem and providing the required responses.

IGL is an interactive color graphics language program with incomplete on-line documentation. To view the construction of an IGL program follow the sequence of commands for

viewing a chart using the RUNFIL "PHOTINT4.RUNFIL". The RUNFIL accesses the subsystem ⟨LEVEL2⟩IGL, specifies the device parameter file, and calls in the IGL graphics file. Tables of color values have been created by LT Ellen Roland, USN, and can be viewed using the following sequence of commands:

(1) cd /usr/cs375Ø/viewgraphs

(2) cat igl.red>/dev/gnt(Ø,1, or 2)

Appendix B is an annotated example IGL program explaining some of the sublevel commands necessary to program in IGL that are not readily apparent to the user in the on-line documentation which can be obtained by exploration.

VIII. CONCLUSIONS

The demonstation game contained in this thesis was played on 23 February, 1981 in the U. S. Naval Postgraduate School C3 Laboratory. Participants were students in the C3 Curriculum representing a wide variety of career skills in the Navy, Air Force, Army, and National Security Agency.

Player reaction to the game was generally favorable. A majority of the issues raised by the players during postgame debriefing would have been resolved in the course of a more comprehensive pregame briefing and training period. It was observed that game play was initially hampered while the players became comfortable with the hardware. Once familiar with the techniques to execute their roles, the players quickly got involved with and began to respond to the scenario.

It was the observation of the authors that this demonstration game provides the means to introduce a wide variety of techniques and technologies that can be used for gaming. However, simpler, less dense scenarios are necessary in order to obtain useful analytic data. The number of players involved in this particular game demonstrated the fact that, altough there are twenty-two access ports to the computer in the laboratory, the computer may not be capable of handling an external software product using all these input/output devices.

In conclusion, it is the authors' opinion that sufficient software and hardware technologies are resident in the C3 Laboratory to provide the tools necessary for research, experimentation and gaming in command and control. Although there are certainly practical limitations that bound the laboratory's capability, they can be overcome by imagination and innovation.

IX. RECOMMENDATIONS FOR FURTHER WORK

It is highly recommended that a dialogue be established between the Command, Control and Communications, Electronic Warfare, Antisubmarine Warfare, and Naval Intelligence curricula to aid in identification of areas of mutual interest that could provide the basis for scenario development, experimentation, and analysis. For example, the C3 Laboratory can provide the environment for an experimental evaluation of the tactical SIGNINT system COMBAT DF. Specifically, force levels, threat levels, and sensor parameters can be defined and varied with WES to provide for iterative results. Utilization of the systems communications capabilities developed in this thesis can provide a realistic organizational and communications structure. Game play in this environment, while not providing detailed system performance data for COMBAT DF, can provide information on force effectiveness with and without this system, the effects of variations in tactical employment, and the effects of variations in organization and communications structures. In this manner, an analysis with the man-in-the-loop can be made of not only COMBAT DF, but of the inherent command, control and communications activities which cannot be represented in a computer-driven simulation.

APPENDIX A

OPERATIONS ORDER FOR CARIBBEAN SCENARIO

File No. ØØ1/81
Exercise Confidential

Caribbean Contingency Forces
TG 25.1 Exercise Group, and
COMCCF
CARON (DD 98Ø), Flagship
At Sea: Enroute Point Option

OPERATION ORDER
COMCCF OPORDER No. 1-81

DTG: 27ØØØ1Z March 1981
Message Ref: SLICK SEVEN

REferences: CINCLANTFLT OPORDER 2Ø1-YR

Zone Time: Use Greenwich Mean Time (ZULU)

Task Organization:

a.	25.1	Caribbean Contingency Force	
		COMCCF and	RADM JONES
		Composite Warfare Commander (CWC)	
		TEXAS (CGN 39)	1 CGN,2 SH3H
		CARON (DD 98Ø), Flagship	1 DD, 1 SH3H
		BROOKE (FFG 1)	1 FFG
		PAUL (FF 1Ø8Ø), BOWEN (FF 1Ø79)	2FF, 2 SH3H
		DETROIT (AOE 4)	1 AOE
b.	25.1.1	Antiair Warfare Commander (AAWC)	
		CO, TEXAS (CGN 39)	
c.	25.1.2	Antisubmarine Warfare Commander (ASWC)	
		CO, PAUL (FF 1Ø8Ø)	
d.	25.1.3	Antisurface Warfare Commander (ASUWC)	
		CO, CARON (DD 98Ø)	
e.	25.1.4	Replenishment Unit	CAPT THOMAS
		DETROIT (AOE 4)	1 AOE
f.	25.1.5	Special Air Support Unit	CDR TOM
		Naval Air Station,	3 P3C, 1 E2C,
		Key West, Florida	1 EP3
g.	25.1.6	Special USAF Support Unit	COMMANDER,
		Homestead Air Force Base,	17TFW
		Florida	

h. 25.1.7 Submarine Service Unit RAY (SSN 653) 1 SSN

i. 25.1.8 Surface Raider Unit PEGASUS (PHM 1), HERCULES (PHM 2) 2 PHM

Operation Order
COMCCF OPORDER 1-81

1. SITUATION. A Soviet task group concluded exercises in the Straits of Florida last week and is presently making a port call in Havana. USS BOWEN, surface tattletale, reported observing participation in the exercises by Cuban maritime patrol aircraft, strike aircraft, and missile patrol boats. Soviet long range strike aircraft also participated in the exercises. Reports have been received from the U.S. Coast Guard and the Federal Aviation Administration that Soviet and Cuban ships and aircraft have been using commercial ships and aircraft as targets for intercepts, tracking exercises, and practice attacks. Activities of this nature in interantional waters and airspace cannot go unopposed.

a. ENEMY FORCES

(1) The Soviet Task Group, consisting of a KRESTA II CG (MAKAROV), KYNDA CG (VARYAG), and two KASHIN DDG's (SMELY, SLAVNY) is inport Havana, Cuba.

(2) At least one and possibly two Soviet submarines are believed to be operating in the Straits of Florida.

(3) Cuban aircraft operating out of airfields near Cienfuegos and Havana consist of 12 FLOGGER strike aircraft and 2 MAY antisubmarine maritime patrol aircraft. 6 BADGER G long range strike aircraft belonging to the Soviet Naval Aviation Force are deployed to Cuba.

b. FRIENDLY FORCES. A special detachment of patrol, early warning, and search aircraft has been formed to support COMCCF operations. The 17TH TACTICAL FIGHTER WING has been tasked to conduct reconnaisance and strike operations for joint coordination and training. PEGASUS and HERCULES have been tasked to conduct surface attacks. RAY will provide submarine services in AREA OPTION.

c. ATTACHEMENTS AND DETACHMENTS. As directed by Commander, Caribbean Contingency Force.

2. MISSION. The Caribbean Contingency Force will conduct joint operations in the Straits of Florida for the purpose of improving coordination and procedures between afloat warfare commanders and multiservice shorebased air assets, and to establish U.S. presence as a deterrent against Soviet or Cuban aggression.

3. EXECUTION. The Caribbean Contingency Force will rendevous at POINT OPTION at 2713ØØZ March 1981 and commence coordinated air, surface, and subsurface exercises. BOWEN will conduct surface tattletale operations in the vicinity of Havana.

a. AAWC

(1) Conduct early warning and task group air defense operations.

(2) Direct employment of TU 25.1.5 AAW asset.

b. ASWC

(1) Conduct air and surface antisubmarine operations.

(2) Direct employment of TU 25.1.5 ASW assets.

c. ASUWC

(1) Conduct task group surface defense.

(2) Direct employment of TU 25.1.5 intelligence assets.

d. TU 25.1.6

(1) Locate and conduct simulated air strikes against TG 25.1 units during the period 2713ØØZ to 2913ØØZ.

e. TU 25.1.8

(1) Locate and conduct simulated missile and gun attacks against TG 25.1 units during the period 2713ØØZ to 2913ØØZ March.

f. COORDINATING INSTRUCTIONS

(1) This order is effective upon receipt.

(2) This is the hurricane season. Be alert for sudden changes in the weather.

(3) Strict adherence to the U.S.-U.S.S.R. Incidents at Sea Agreement is imperative.

(4) Direct liaison between warfare commanders info CWC, is authorized.

(5) Direct inquiry to DIA is authorized.

4. ADMINISTRATION AND LOGISTICS. Report any mission degrading deficiencies to CWC as necessary.

5. COMMAND AND SIGNAL.

a. Use effective CCF Communication Plan.

b. AAWC is second in command.

c. Commander Caribbean Contingency Force, Officer in Tactical Command.

Accknowledgement Instruction. Task Unit Commanders acknowledge receipt by message using message reference number.

T. T. JONES
CTG 25.1 and COMCCF

Authentication:

T. A. CAMO
Commander, U.S. Navy
Flag Secretary

ANNEX C: CCF Communications Plan
ANNEX M: Game Message Sequence
ANNEX N: NMCC Player Instructions
ANNEX U: EXSUP/ORANGE Player Instructions

ANNEX CHARLIE

CCF Communications Plan

PLAYER	MESSAGE ADDRESS
National Command Authority	NCA
Joint Chiefs of Staff	JCS
Commander-in-Chief, U.S. Atlantic Fleet	CLANFLT
Graphics Support	NMCC
Commander 17th Tactical Fighter Wing	TFW17
Defense Intelligence Agency	DIA
Commander, Caribbean Contingency Force	COMCCF
Composite Warfare Commander	CWC
Antiair Warfare Commander	AAWC
Antisubmarine Warface Commander	ASWC
Antisurface Warfare Commander	ASUWC

ANNEX MIKE

Scenario Messages

MESSAGE1

2713ØØZ MAR 81

FM FOSIC NORFOLK, VA
TO COMCCF

BT
EXERCISE CONFIDENTIAL
CARIBBEAN INTELLIGENCE SUMMARY

1. A SOVIET TASK GROUP, CONSISTING OF A KRESTA II CG, KYNDA CG, AND TWO KASHIN DDG'S IS INPORT HAVANA. THERE HAS BEEN NO AIR ACTIVITY IN THE PAST 24 HOURS.

2. THERE HAS BEEN NO CONFIRMATION OF SOVIET SUBMARINE PARTICIPATION IN RECENT SOVIET-CUBAN EXERCISES. ANALYSIS OF THE EXERCISES INDICATES THAT THERE WAS AT LEAST ONE AND POSSIBLY TWO SUBMARINES INVOLVED.
BT

MESSAGE2

2713Ø5Z MAR 81

FM USS BOWEN
TO ASUWC
INFO CWC

BT
EXERCISE CONFIDENTIAL
SOVIET TASK GROUP SORTIE

1. SOVIET TG UNDERWAY. COURSE Ø45 SPEED 25. THIS UNIT REMAINING 5 REPEAT 5 MILES ABEAM.
BT

MESSAGE3

271310Z MAR 81

FM DIA
TO JCS

BT
EXERCISE CONFIDENTIAL
SPOT INTELL REPORT - CUBA

1. THE SOVIET TASK GROUP PRESENTLY INPORT HAVANA HAS EXCEEDED THE PREVIOUS RECORD LENGTH PORT CALL. IT IS ESTIMATED THAT THESE UNITS WILL SAIL FOR HOME WATERS IN THE NEXT TWENTY-FOUR HOURS.

2. THE SOVIET AMBASSADOR TO CUBA ANNOUNCED THE SUCCESSFUL CONCLUSION OF JOINT SOVIET-CUBAN EXERCISES AND WARMLY PRAISED THE PROFICIENCY AND CAPABILITIES OF THE SOVIET UNION'S "STAUNCH AND VALUED ALLY". A PROMISE OF FURTHER SUPPORT AND MODERNIZATION OF CUBA'S FORCES WAS MADE.
BT

MESSAGE4

271315Z MAR 81

FM FOSIC NORVA
TO COMCCF
INFO CLANFLT

BT
EXERCISE SECRET
BADGER WARNING ADVISORY

1. ONE BADGER LAUNCHED FROM MANZANILLO AT 271310Z ON COURSE 355 SPEED 450.
BT

MESSAGE5

271325Z MAR 81

FM FNOC NORFOLK, VA
TO COMCCF
INFO CLANFLT

BT
EXERCISE CONFIDENTIAL
48 HOUR WEATHER FORECAST

1. HURRICANE ANDY, CENTERED AT 2ØN 66E CONTINUES TO TRACK NORTHWEST AT 1ØKTS. WINDS OVER 1ØØKTS EXTEND OUT 3ØØ MILES FROM THE EYE.

2. THE FORECAST FOR THE PERIOD 2713ØØZ TO 2913ØØZ IN COMCCF AREA OF OPERATIONS CALLS FOR AN OVERCAST SKYS WITH THE CEILING OCCASIONALLY DROPPING TO UNDER 1ØØØFT, WINDS FROM THE SOUTHWEST AT 1ØKTS GUSTING TO 25KTS INCREASING TO 23-3ØKTS WITH GUSTS UP TO 5ØKTS POSSIBLE BY 28Ø8ØØZ, AND SEAS 6-8FT INCREASING TO 15-2ØFT AS THE WIND INCREASES.
BT

MESSAGE6

27133ØZ MAR 81

FM FOSIC NORFOLK, VA
TO COMCCF
INFO CLANFLT

BT
EXERCISE SECRET
MAY WARNING ADVISORY

1. ONE MAY LAUNCHED AT 2713Ø8Z FM CIENFUEGOS COURSE Ø3Ø SPEED 35Ø.
BT

MESSAGE7

2714ØØZ MAR 81

FM NCA
TO JCS

BT
EXERCISE SECRET
SOVIET CARIBBEAN NAVAL ACTIVITY

1. IN VIEW OF COMPLAINTS RECEIVED DURING THE SOVIET-CUBAN EXERCISES FROM MERCHANT SHIPPING AND COMMERCIAL AIRLINES CANCEL THE CARIBBEAN CONTINGENCY FORCE EXERCISES AND DIRECT COMCCF TO CONDUCT DISCREET SURVEILLANCE OF THE SOVIET TASK GROUP UNTIL THEY PASS INTO THE ATLANTIC.
BT

MESSAGE8

27142ØZ MAR 81

FM   USS BOWEN
TO   ASUWC
INFO CWC

BT
EXERCISE CONFIDENTIAL
SOVIET SITREP

1. AT 134Ø OBSERVED SOVIET TASK GROUP SPLIT INTO TWO SECTIONS. KRESTA II AND KASHIN REMAINED ON COURSE Ø45 SPEED 25. KYNDA AND KASHIN CHANGED COURSE TO Ø9Ø SPEED 25.

2. SHORTLY AFTER THE TASK GORUP SPLIT UP BOWEN SUFFERED A HIGH WATER CASUALTY TO 1A BOILER AND LOSS OF POWER. VISUAL CONTACT WITH SOVIET TASK GROUP LOST IN RAIN SQUALL.

3. READY FOR DUTY IN ALL RESPECTS.
BT

MESSAGE9

271430Z MAR 81

FM NCA
TO JCS

BT
EXERCISE SECRET
CARIBBEAN MERCHANT SHIPPING

1. DIRECT COMCCF TO ASCERTAIN IDENTITY AND NATIONALITY OF ALL MERCHANT SHIPPING IN THE STRAITS OF FLORIDA AS SOON AS POSSIBLE.

2. WHENEVER SOVIET OR COMMUNIST BLOC SHIPPING IS LOCATED, INFORM THE NCA IMMEDIATELY.
BT

MESSAGE10

271435Z MAR 81

FM FOSIC NORFOLK, VA
TO COMCCF
INFO CLANFLT

BT
EXERCISE SECRET
BADGER WARNING ADVISORY

1. ONE BADGER LAUNCHED FROM MANZANILLO AT 271415Z ON COURSE 355 SPEED 450.
BT

MESSAGE11

271445Z MAR 81

FM TFW17
TO CWC

BT
EXERCISE CONFIDENTIAL
THUNDERSTORM ACTIVITY

1. HEAVY THUNDERSTOM CELL DIRECTLY OVER HOMESTEAD PRECLUDES LAUNCHES AT THIS TIME. EXPECT TO BE ABLE TO RESUME OPERATIONS IN THIRTY MINUTES.
BT

MESSAGE12

271455Z MAR 81

FM TFW17
TO CWC

BT
EXERCISE CONFIDENTIAL
READY STATUS REPORT

1. MAIN THUNDERSTORM ACTIVITY HAS CLEARED. READY TO RESUME OPERATIONS AT THIS TIME. MINOR CELLS ACTIVITY EXPECTED TO HAVE MINIMUM IMPACT ON OPERATIONS.
BT

MESSAGE13

2715Ø5Z MAR 81

FM DIA
TO JCS

BT
EXERCISE SECRET
SPOT INTELLIGENCE REPORT - CUBA

1. INFORMED SOURCES HAVE REPORTED MAJOR CONSTRUCTION WORK BEING CARRIED OUT UNDER STRICT SECURITY IN HEAVILY WOODED REGION NEAR CIENFUEGOS BY RUSSIAN COMBAT ENGINEERS AND CIVILIAN TECHNICIANS.
BT

MESSAGE 14

271515Z MAR 81

FM NCA
TO JCS

BT
EXERCISE SECRET
THREAT ASSESSMENT

1. NATIONAL SECURITY COUNCIL ADVISORS ASSESS CURRENT ACTIVITY IN THE CARIBBEAN AS AN ATTEMPT BY THE SOVIET UNION TO INTRODUCE OFFENSIVE NUCLEAR WEAPONS INTO CUBA.

2. FAILURE OF THE U.S. INTELLIGENCE COMMUNITY TO PROVIDE ADEQUATE INDICATIONS AND WARNING, AND THE SHORT TIME AVAILABLE TO PREVENT A FAIT ACCOMPLI, PUTS THE U.S. IN A VERY DELICATE POSITION.

3. JCS SITUATION ASSESSMENT AND OPTIONS AVAILABLE REQUIRED ASAP.
BT

MESSAGE15

271525Z MAR 81

FM NCA
TO JCS

BT
EXERCISE SECRET
BLOCKADE INSTRUCTIONS

1. UNTIL SUCH TIME AS ALL FEASIBLE ALTERNATIVES CAN BE EXPLORED, AND DIPLOMATIC OPTIONS EXHAUSTED, DIRECT COMCCF TO POSITION HIS FORCES IN SUCH A MANNER THAT THERE IS NO DOUBT OF U.S. INTENT TO PREVENT SOVIET MERSHIPS FROM ENTERING PORT.

2. THE CCF SHOULD BE IN A FULLY COMBAT READY STATE. REPORT ANY DEFICIENCIES THAT WOULD PRECLUDE OPERATIONS UNDER COMBAT CONDITIONS.
BT

MESSAGE16

271535Z MAR 81

FM FNOC NORFOLK, VA
TO COMCCF
INFO CLANFLT

BT
EXERCISE CONFIDENTIAL
HURRICANE ADVISORY

1. PROJECTION OF THE TRACK OF HURRICANE ANDY INDICATES PASSAGE THROUGH THE STRAITS OF FLORIDA IN THE EARLY MORNING HOURS OF 28 MAR.

2. THE UNUSUALLY STEADY TRACK OF THIS HURRICANE SUGGESTS SLIGHT POSSIBILITY OF DEVIATION.
BT

MESSAGE17

271540Z MAR 81

FM NCA
TO JCS

BT
EXERCISE SECRET
CCF CAPABILITIES

1. ARE U.S. FORCES AVAILABLE TO COMCCF CAPABLE OF CONDUCTING BLOCKADE OPERATIONS IF ACTIVELY OPPOSED BY THE SOVIET TASK GROUP.
BT

MESSAGE18

271555Z MAR 81

FM NCA
TO JCS

BT
EXERCISE SECRET
CCF FULES OF ENGAGEMENT

1. DIRECT COMCCF TO ENFORCE BLOCKADE OF SOVIET MERCHANT SHIPPING TO CUBA BY EVERY MEANS AT HIS DISPOSAL.

2. ACTIVE INTERFERENCE BY THE SOVIET TASK GROUP IS CONSIDERED JUSTIFICATION FOR AN ALL OUT RETALIATORY STRIKE.
BT

MESSAGE19

2716ØØZ MAR 81

FM USS PEGASUS
TO ASUWC
INFO CWC

BT
UNCLAS
MISSILE ATTACK

1. THIS UNIT WAS FIRED UPON BY CUBAN PGM'S MINOR DAMAGE INCURRED TO GUN MOUNT. PGM'S ARE RETREATING NORTHWEST AT HIGH SPEED. REQUEST INSTRUCTIONS.
BT

ANNEX NOVEMBER

NMCC Player Instructions

1. The function of the NMCC is to provide graphics support to JCS and CLANFLT. After logging into NMCC type the following:

(1) sh<maprunfilØ

(2) telnet tops2Ø

(3) log(esc)nps3(esc)password(esc esc)

(4) runfil

The system will inquire for the name of the RUNFIL to be executed. The sequence of RUNFIL to be executed is as follows:

TIME	RUNFIL NAME
13ØØ	INIT1.RUNFIL
13Ø5	CCF1.RUNFIL
131Ø	CCF2.RUNFIL
133Ø	OSIS2.RUNFIL
143Ø	OSIS3.RUNFIL
1435	PHOTINT1.RUNFIL
145Ø	PHOTINT2.RUNFIL
15Ø5	PHOTINT3.RUNFIL
152Ø	PHOTINT4.RUNFIL

Honor message requests from JCS and CLANFLT, in that order, to review graphics files already shown. Start a new RUNFIL at the time indicated. Wait for the system to respond with "FINISHED" before starting a new RUNFIL.

ANNEX UNIFORM

EXSUP/ORANGE Instructions

NO.	TIME	EVENT
1	12ØØ	Briefing/Oporder review.
2	123Ø	Oporder Acknowledgement/Communications Check.
3	13ØØ	Start WES/Start Message Generator.
4	13ØØ	Launch 2 RF4 for reconnaisance, CSE 14Ø SPD 3ØØ ALT 2Ø,ØØØ.
5	13ØØ	Soviet Sortie from Havana, CSE Ø43 SPD 25.
6	131Ø	Launch BADGER for area round robin, CSE 355 SPD 45Ø ALT 1Ø,ØØØ, SENSORS ON.
7	1315	Emit from JULIETT for 2 minutes.
8	1315	Launch MAY for sonobouy work, CSE Ø3Ø SPD 35Ø ALT 5ØØØ SENSORS ON.
9	1325	Maneuver CHARLIE to provide a detection for BOWEN.
1Ø	1335	Launch 4 F4E for simulated strike on CCF, CSE 14Ø SPE 4ØØ ALT 15,ØØØ SENSORS ON.
11	134Ø	Split Soviet TG into 2 SAG's, KYNDA and KASHIN CSE Ø45 SPD 25, KRESTA II and KASHIN CSE Ø9Ø SPD 25.
12	1345	BOWEN casualty. DIW with no sensors.
13	14Ø5	Emit from JULIETT for 2 minutes.
14	141Ø	Maneuver JULIETT to provide a detection for CCF.
15	1415	Launch BADGER for area round robin, CSE 355 SPD 45Ø ALT 1Ø,ØØØ SENSORS ON.

16	142Ø	Restore BOWEN, CSE Ø45 SPD 2Ø SENSORS ON.
17	1425	Emit from one Soviet SAG.
18	144Ø	Manuever PGM's in the direction of PEGASUS and HERCULES.
19	145Ø	Emit from second Soviet SAG.
2Ø	151Ø	Launch 2 BADGERS at CCF, CSE 36Ø SPD 45Ø ALT 5ØØ SENSORS ON.
21	152Ø	Put Soviet SAG's on rendevous course with Soviet merchant ships.
22	153Ø	Emit from JULIETT for 2 minutes.
23	1545	Launch 6 FLOGGERS, CSE 36Ø SPD 4ØØ ALT 1ØØØ SENSORS OFF.
24	1550	Launch 4 BADGERS, CSE 36Ø SPD 4ØØ ALT 5ØØ SENSORS OFF.
25	16ØØ	Cuban PGM's launch missiles at PEGASUS.
26	16ØØ	Launch all missiles from JULIETT at CCF.
27	16ØØ+	HOT WAR

From 16ØØ on an interactive umpire/player exchange in a localized warfare environment will determine the direction of the game.

APPENDIX B

FIGURES

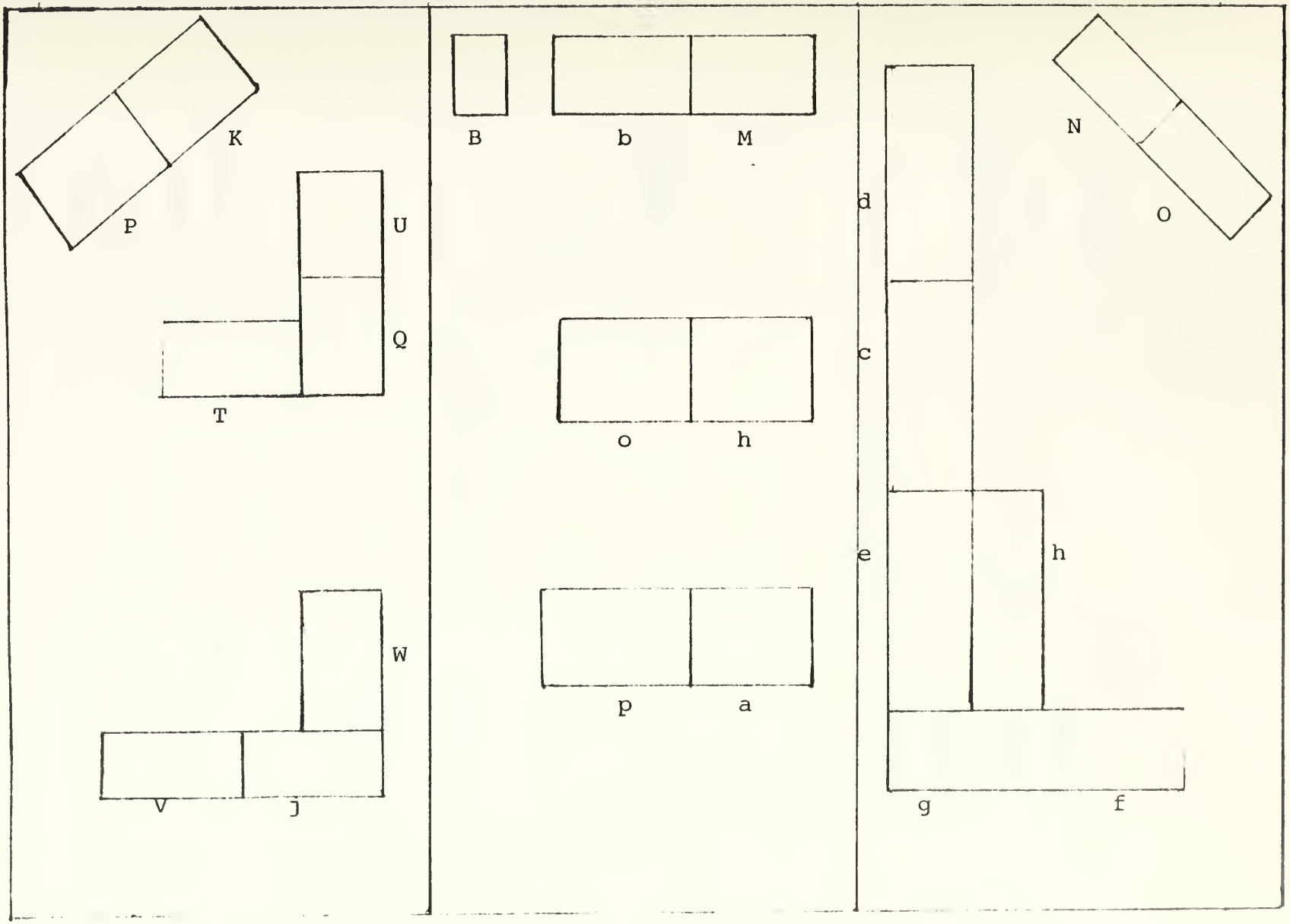


FIGURE 1. C3 LABORATORY

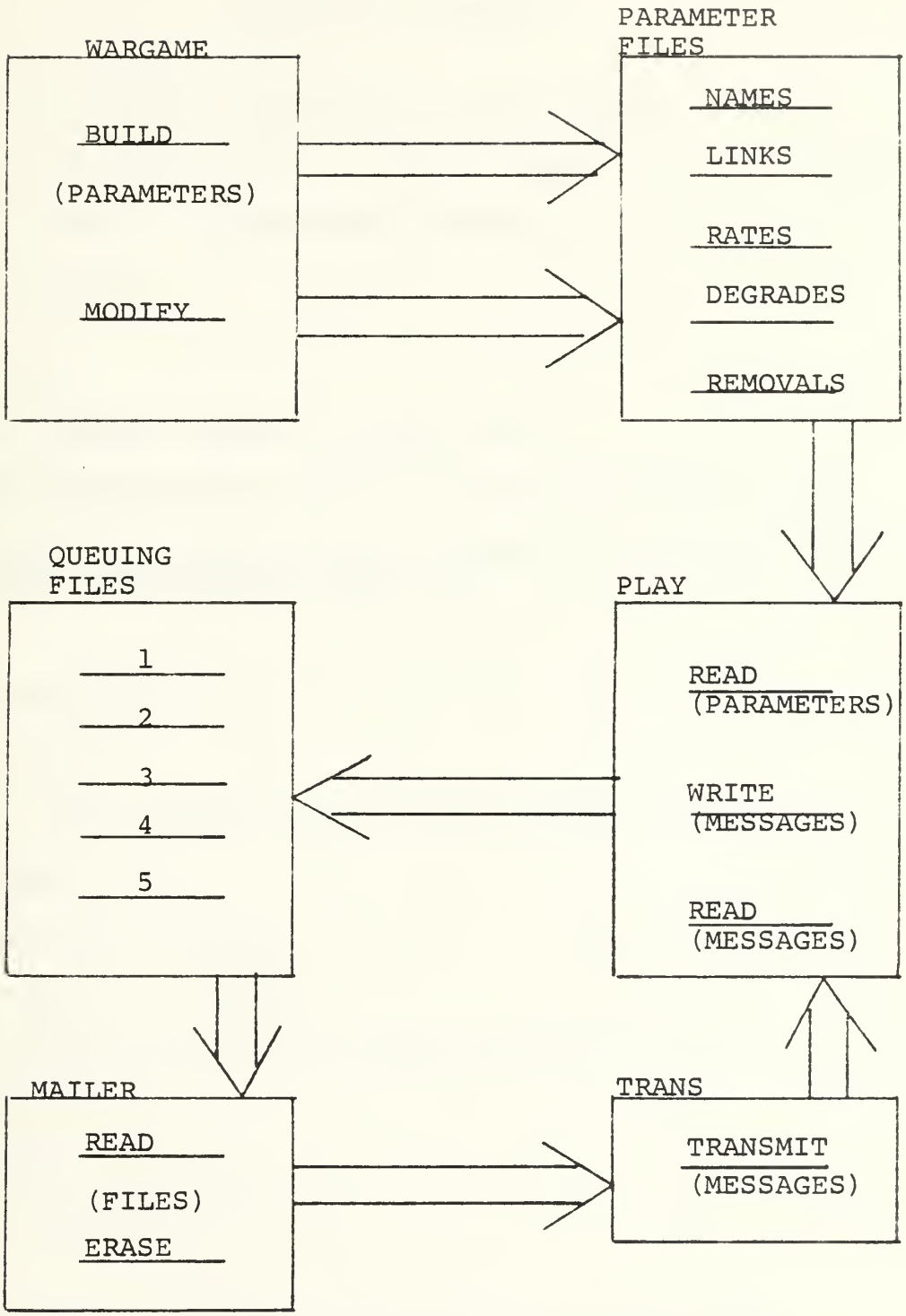


FIGURE 2. FLOW DIAGRAM FOR WARGAME.C, PLAY.C AND MAILER.C

APPENDIX C

SAMPLE IGL PROGRAM

The IGL instructions contained in this appendix are two representative ports from a graphics RUNFIL at TOPS20 called "PHOTINT1.RUNFIL." Explanatory comments follow each line of code in brackets.

```
⟨LEVEL2⟩ IGL
[Accesses sublevel LEVEL2, program IGL]
INIT "BACKEND=(GENISCO),DEVICE=(S,NET:Ø.NPS-UNIX-77;T,Ø)"
[Device parameter file defining computer to computer path from
ACCAT TOPS2Ø to NPS UNIX, Genisco Ø]
OPEN 1
[Opens first port]
COLOR 1 Ø Ø 1
[Intensity, Red, Green, Blue; values from Ø to 1.Ø]
FILL-POLYGON
[Prepare to draw polygon]
VERTEX Ø Ø Ø 1 1 1 1 Ø Ø Ø
[Define vertices of a polygon from start back to start; values
from Ø to 1.Ø]
END-POLYGON
[Ends drawing polygon]
CLOSE
[Closes port 1; a CLOSE is required from every OPEN]
POST
[Executes commands in port]
```

```
OPEN 6

[Opens port 6; ports must be in numerical sequence]

COLOR 1 .2 Ø Ø

[Same as port 1]

FACE

[Prepare to accept text typeface instructions]

HE .Ø8

[Height of typeface is to be .Ø8; values from Ø to 1.Ø]

WI .Ø4

[Width of typeface is to be .Ø4; values from Ø to 1.Ø]

Q 1

[Quality of typeface is to be 1.Ø; no other values]

NAME 2

[Style of typeface is to be 2; 1=English, 2=Cyrillic,
3=Greek]

MOVE .2 .32

[Prepares to draw at position .2 .32 (X-Y); values from
Ø to 1.Ø]

TEXT "LENIN"

[Text to be drawn is "LENIN"]

CLOSE

[As before]

POST

[As before]
```

THE MESSAGE HANDLING MODULE

The message handling module consist of four main programs. The code for three of them is contained in this document: wargame.c, play.c and mailer.c. The fourth, trans.c, is an adaptation of the UNIX SNDMSG subsystem. The code for this program is contained in the directory "control" residing on the PDP 11/70 in the C3 Laboratory at the Naval Postgraduate School.

```
/*    WARGAME HAS THE FOLLOWING MAJOR SUBPROGRAMS:

          1.  BUILD
              LAB FLOORPLAN
              ORGANIZATIONAL STRUCTURE
              COMMUNICATIONS STRUCTURE/PARAMETERS

          2.  MODIFY (during play)
              COMMUNICATIONS PARAMETERS
              GAME PLAYERS


      THE FOLLOWING OTHER PROGRAMS ARE ALLIED WITH WARGAME

                PLAY
                MAILER
                TRANS

                 1.  PLAY
                     MESSAGE RECEIPT
                     MESSAGE CONSTRUCTION
                     MESSAGE CORRUPTION CALCULATIONS
                     DATA COLLECTION
                     SIMULTANEOUS GAMES

                 2.  MAILER
                     ASYNCHRONOUS PROCESS WITH PLAY
                     READS MESSAGE FILES WRITTEN BY PLAY

                 3.  TRANS
                     MODIFICATION OF BBN SNDMSG
                     SENDS MESSAGES READ BY MAILER
*/
```

```
char nameidx[20][20],clear 014,aray[10],action[20],from[20],pswd[10];
char deadidx[20][20],anam[10];
int atoi(),matrix[20][20], rand(),qty;
float atof(),wq;
double log();

main(){
  char a; int i, n, comp;
  printf("%c0,clear);
  clock();
  printf("                                   WARGAME0);

    printf("        WHAT PORTION OF WARGAME DO YOU WISH TO ACCESS?0);
    printf("             BUILD  (B) 0);
    printf("             MODIFY (M) 0);
    printf("             PLAY   (P) 0);
    anam[0]=' ';anam[1]=' ';anam[2]=' ';anam[3]=' ';anam[4]=' ';
    anam[5]=' '; anam[9]=' ';                  /* Note that the internal */
    while(1){                                   /* password values have   */
    a=getans(a);                                /* been removed from the  */
    printf("%c0,clear);                    /* uncompiled code.       */
    switch(a){
      case'b':case'B':
       printf("0);
       printf("            PLEASE ENTER THE WARGAME PASSWORD0);
       for(n=0;n<10&&(pswd[n]=getchar())!='0;n=n+1);pswd[n]=' ';
       n=0;comp=0;
       while(comp==0&&n<10){comp=scmp(pswd[n],anam[n]);n++;}/*psswd ck*/
         if(n==10) build();
         else{printf("%c0,clear);
       printf("       DO NOT TRY TO ENTER 'BUILD' UNLESS AUTHORIZED0);}
       sleep(5);return;
       break;
      case'p':case'P':
       play();
```

```
        break;
       case'm':case'M':
        printf("Ø);
        printf("          PLEASE ENTER THE WARGAME PASSWORDØ);
        for(n=Ø;n<1Ø&&(pswd[n]=getchar())!='Ø;n=n+1);pswd[n]=' ';
        n=Ø;comp=Ø;
        while(comp==Ø&&n<1Ø){comp=scmp(pswd[n],anam[n]);n++;}
          if(n==1Ø) modify();                          /* Password check */
          else{printf("%cØ,clear);
          printf("    DO NOT TRY TO ENTER 'MODIFY' UNLESS AUTHORIZEDØ);}
        sleep(5);return;
        break;
       case'q':case'Q':
        stop();
        break;
       default:
        printf("TYPE (B)uild, (M)odify, (P)lay or (Q)uit   ONLYØ);
        break;

    }
  }
}

/*-------------------------------------------------------------------*/


/*  CHARACTER FETCHING SUBROUTINE   */

getans(m){
  char ray[1Ø];
  int i;
  for(i=Ø;(m=getchar())!='Ø;i=i+1) ray[i]=m;
  m=ray[Ø];
  return(m);
}
```

```
/*  PROGRAM EXIT SUBROUTINE  */

stop(){
  char b;
  printf("          CONFIRM YOU WISH TO EXIT THE PROGRAM. (Y/N)Ø);
  b=getans(b);
  if(b!='y') return;
  printf("                THANK YOU AND GOODBYEØ);
  exit();
}

/*  TIME PRINTING SUBROUTINE  */

clock(){
  int tvec[2],time(),i,*localtime(),*locl;
  char *ctime();
  time(tvec);
  printf("ØHE TIME IS: %s",ctime(tvec));
  printf("Ø);
}


/* ----------------------------------------------------------------*/

/*  BUILD   */

build(){
  char c;
  int fk;
  printf("
        THIS PORTION OF THE PROGRAM WILL ALLOW THE UMPIRE TO:

         1. REVIEW THE LAB FLOORPLAN (F)

         2. DESIGN THE ORGANIZATIONAL STRUCTURE FOR THE GAME (O)
```

```
            3. DESIGN THE COMM NET SUPPORTING THE ORGANIZATION  (C)0);

  while(1){
    c=getans(c);
    switch(c){
      case'o':case'O':
        if((fk=fork())==0){           /* Fork to organization display.*/
         execl("/usr/cntrlf/diagram1","/usr/cntrlf/diagram1",0);
         exit();}
        wait(&fk);
        printf("TYPE 'F' FOR THE FLOOPLAN, 'C' FOR THE COMM NETWORK0);
        break;
      case'f':case'F':
        if((fk=fork())==0){             /* Fork to floorplan display. */
         execl("/usr/cntrlf/floorplan","/usr/cntrlf/floorplan",0);
         exit();}
        wait(&fk);
        printf("TYPE 'O' FOR ORG STRUCTURE, 'C' FOR THE COMM NET0);
        break;
      case'c':case'C':
        comm();
        break;
      case'q':case'Q':
        stop();
        break;
      default:
        printf("TYPE O, C, F, OR Q ONLY0);
        break;
     }
   }
}

/*  INTERCOMMUNICATIONS MATRIX SUBROUTINE  */

comm(){
  char a,players[3],b[2]; int f,i,j,k,n,m;
```

```
printf("%c0,clear);

printf("HOW MANY PLAYERS, INCLUDING THE UMPIRE, ARE THERE?0);
f=open("/usr/cntrlf/number",2);
for(m=0;m<3&&(players[m]=getchar())!='0;m++); players[m]=' ';
write(f,&players[0],4);
close(f);
qty=atoi(players);
printf("          THE NUMBER OF PLAYERS IS:  %d0,qty);
if(qty>20){printf("ONLY 20 PLAYING POSITIONS PER SESSION!0);comm();}

for(i=1;i<(qty+1);i++) {              /* Initialize the communications */
 for(j=1;j<(qty+1);j++)               /* matrix to zero.               */
   matrix[i][j]=0; }

printf("WHAT ARE THE NAMES OF THE PLAYERS IN THE GAME? (no caps) 0);
f=open("/usr/cntrlf/index",2);        /* Write the player's names and  */
i=j=0;                                /* load them into the file index */
while(i<qty){
  i++;
  for(n=0;n<20&&(nameidx[i][n]=getchar())!='0;n++);
  nameidx[i][n]=' ';
  printf("                        Player #%d is:  %s0,i,&nameidx[i][0]);
  write(f,&nameidx[i][0],20);}
close(f);

printf("   0ou have named %d players0,qty);
sleep(4);
printf("%c0,clear);
printf("
        If you wish ALL circuits to have same value and ALL players
        to be able to communicate with ALL OTHER players directly,
        TYPE 'all'.

        OTHERWISE, TYPE <cr>
```

```
printf("
        ENTER THE CIRCUIT QUALITY FOR EACH PAIR OF PLAYERS
        AS SHOWN IN THE LEGEND BELOW:

             0 -- None (No Communications Link)
             1 -- Encrypted Landline with Precedence Control
             2 -- Non-Encrypted Landline (AUTODIN)
             3 -- Digital RF Circuits (HF/VHF/UHF) with A/J
             4 -- Digital RF Circuits without A/J protection
             5 -- Analog Voice
             6 -- Perfect Link (No Degrade, No Delay)

        ALL COMM LINKS HAVE BEEN PRESET TO 0.  SELECT THE VALUE
        REQUIRED  AS THE CHOICE IS PRESENTED TO YOU PLUS <cr>
        (If 0 is the correct value, type <cr> only.)

a=getans(a);
if((a=='a')||(a=='A')){
f=open("/usr/cntrlf/matrix",2);
for(i=1;i<(qty+1);i++) {              /* Set the values of the comm    */
 for(j=1;j<(qty+1);j++){              /* matrix to 'no degrade'.       */
   matrix[i][j]=6;
   write(f,"6",2);} }
}
else{

f=open("/usr/cntrlf/matrix",2);    /* Write the comm qualities into  */
for(i=1;i<(qty+1);i++) {           /* the 2-d array 'matrix' and load*/
 for(j=1;j<(qty+1);j++) {          /* the array into the file'matrix'*/
   if(matrix[i][j]>0) break;
   if(i==j) break;
   printf("0etween %s and %s is: %d0,&nameidx[i][0],
   &nameidx[j][0],matrix[i][j]);
   for(n=0;n<2&&(b[n]=getchar())!='0;n++);b[n]=' ';
   write(f,&b[0],2);
```

```
      k=atoi(b);
      if(k>6){printf("THE VALUE %s IS NOT ACCEPTABLE\n",&b[0]);
              {printf("   The circuit value between %s and %s is:%d\n",
              &nameidx[i][0],&nameidx[j][0],matrix[i][j]); }
              for(n=0;n<2&&(b[n]=getchar())!='\n';n++);b[n]=' ';
              write(f,&b[0],2);
              k=atoi(b); matrix[i][j]=k; matrix[j][i]=k;
              printf("            The value is now: %d\n",matrix[i][j]);}
      else{ matrix[i][j]=k; matrix[j][i]=k;
            printf("            The value is now: %d\n",matrix[i][j]);}
    }
  }
  close(f);
  }
  printf("RETURN TO THE MAIN MENU BY DEPRESSING <cr>\n");
  a=getans(a);
  main();
}

/*  MESSAGE PARAMETER INITIALIZATION AND MODIFICATION SUBROUTINE  */

modify(){
  char value[8],loss[5],garbage[5],getm[2],getl[2],chloss[5],chgarb[5];
  char pick1[5],pick1a[5],pick2[5],pick2a[5],pick3[5],pick3a[5];
  char pick4[5],pick4a[5],pick5[5],pick5a[5],count[2],getg[2];
  char a;
  int f,g,i,k,l,m,n;

  printf("%c\n",clear);
  printf("THIS PORTION OF THE PROGRAM ALLOWS THE UMPIRE TO:

              1.  Choose which message arrival and message service
                  rates he wishes to use in the game.
              2.  Establish initial values for these rates.
              3.  Establish the frequency for lost messages.
              4.  Establish the frequency for garbled messages.
```

5. Remove players from the game.

Based on the queuing algorithm for single server facilities the average amount of transmission delay for messages addressed to a given facility (WQ) can be expressed as a function of the average message arrival rate (A) and the average meassage service rate (S). WQ = A/S(S-A)

1. You may specify the actual arrival rates and service rates for each class of communications (1 through 5) or you may rely upon a pre-established service rate/arrival rate relationship and vary only the message arrival rate by requesting:

a. Normal Traffic
b. Medium Traffic (twice the normal arrival rate)
c. Heavy Traffic (three times the norral rate)

TYPE '1' TO INSERT SPECIFIC ARRIVAL RATES and SERVICE RATES
TYPE '2' TO USE THE GENERAL RATES (Normal,Medium,Heavy)

Note: If you do not wish to change values currently set, TYPE <cr>

```
f=open("/usr/control/msgrate",2);          /*Load the message rate */
for(n=0;n<2&&(getm[n]=getchar())!='2';n++);/*decision into the file*/
if(getm[0]=='2') {close(f);goto three;}    /*named 'msgrate'      */
getm[n]=' ';
write(f,&getm[0],4);
close(f);
printf("%c",clear);


if(getm[0]=='2') {close(f);goto three;}
if(getm[0]=='1'){
  printf("
        TO PREVENT A QUEUE WHICH GROWS WITHOUT BOUND, INSURE A/S<1
```

```
        Note:  Message rates are in numbers of messages/minute.
               Use real numbers of 99.99 or less.
printf("For circuit quality 1: (Encrypted Landline)0);
  printf("         A=");
f=open("/usr/cntrlf/pick1",2);
  for(n=0;n<5&&(pick1[n]=getchar())!='0;n++);
  write(f,&pick1[0],4);                  /*  Load each message arrival*/
  printf("         S=");                 /*  and service rate into the*/
f=open("/usr/cntrlf/pick1a",2);         /*  files 'pick()/pick()a.   */
  for(n=0;n<5&&(pick1a[n]=getchar())!='0;n++);
  write(f,&pick1a[0],4);
close(f);

printf("For circuit quality 2: (Non-Encrypted Landline)0);
  printf("         A=");
f=open("/usr/cntrlf/pick2",2);
  for(n=0;n<5&&(pick2[n]=getchar())!='0;n++);
  write(f,&pick2[0],4);
  printf("         S=");
f=open("/usr/cntrlf/pick2a",2);
  for(n=0;n<5&&(pick2a[n]=getchar())!='0;n++);
  write(f,&pick2a[0],4);
close(f);

printf("For circuit quality 3: (Digital RF A/J Resistant)0);
  printf("         A=");
f=open("/usr/cntrlf/pick3",2);
  for(n=0;n<5&&(pick3[n]=getchar())!='0;n++);
  write(f,&pick3[0],4);
  printf("         S=");
f=open("/usr/cntrlf/pick3a",2);
  for(n=0;n<5&&(pick3a[n]=getchar())!='0;n++);
  write(f,&pick3a[0],4);
close(f);

printf("For circuit quality 4: (Digital RF No A/J)0);
```

```
    printf("          A=");
  f=open("/usr/cntrlf/pick4",2);
    for(n=0;n<5&&(pick4[n]=getchar())!='0';n++);
    write(f,&pick4[0],4);
  printf("          S=");
  f=open("/usr/cntrlf/pick4a",2);
    for(n=0;n<5&&(pick4a[n]=getchar())!='0';n++);
    write(f,&pick4a[0],4);
  close(f);

  printf("For circuit quality 5: (Plain Voice)0);
    printf("          A=");
  f=open("/usr/cntrlf/pick5",2);
    for(n=0;n<5&&(pick5[n]=getchar())!='0';n++);
    write(f,&pick5[0],4);
  printf("          S=");
  f=open("/usr/cntrlf/pick5a",2);
    for(n=0;n<5&&(pick5a[n]=getchar())!='0';n++);
    write(f,&pick5a[0],4);
  close(f);

printf("
        Changes to these values which are desired during the game
        should be made by re-entering 'Modify' or by editing files
        named 'pick1''pick1a' through 'pick5''pick5a' respectively
        directory named 'cntrlf' using the password 'wargame'.f);
}

if(getm[0]=='2'){
  printf("
        The pre-established relationship between arrival rates and
        service rates for a NORMAL arrival rate is as follows:
              For circuit quality 1:  S = 3.10A
              For circuit quality 2:  S = 3.05A
              For circuit quality 3:  S = 3.04A
              For circuit quality 4:  S = 3.03A
```

```
                    For circuit quality 5:  S = 3.02A

      printf("Enter 'normal', 'medium' or 'heavy' to establish the
      printf("initial message arrival rate.\0);
      f=open("/usr/cntrlf/prerate",2);
        for(n=0;n<8&&(value[n]=getchar())!='\0';n++);
        write(f,&value[0],8);
      close(f);
    printf("
            Changes to these values which are desired during the game
            should be made by re-entering 'Modify' or by editing file
            named 'prerate' in the directory named 'cntrlf' using
            the passwcrd 'wargame'.

    }
    printf("       TO CONTINUE, DEPRESS <cr>\0);
    a=getans(a);

three:

    printf("%c\0,clear);
    printf("
            3.  Enter the rate for messages to be LOST.  Five equals
            five percent.  Use integer values.

            TYPE '1' IF YOU WISH TO ENTER A SEPARATE LOSS RATE FOR
            EACH CIRCUIT TYPE.  (1-5)

            TYPE '2' IF YOU WANT A STANDARD RATE FOR ALL CIRCUITS

            Note:  If you do not wish to change values currently set
                   TYPE <cr>

    f=open("/usr/control/lossrate",2);      /* Load the message loss rate*/
    for(n=0;n<2&&(getl[n]=getchar())!='\0';n++); /* decision into the   */
    if(getl[0]=='\0'){close(f);goto four;} /*  named 'lossrate'.        */
```

```
    getl[n]=' ';
    write(f,&getl[0],4);
    close(f);

    if(getl[0]=='1') {
      printf("     ENTER THE LOSS RATES BY COMM CIRCUIT QUALITY   \n");

      f=open("/usr/control/loss1",2);
      i=0;                                    /*Load the message lost rates*/
      while(i<5){                             /*for each circuit quality.  */
        i++;
        printf("\nFor circuit quality %d:\n",i);
        printf("          Lossrate =");
        for(n=0;n<5&&(chloss[n]=getchar())!='\n';n++); chloss[n]=' ';
        write(f,&chloss[0],4);}
      close(f); }

    if(getl[0]=='2'){
      printf("     ENTER THE STANDARD RATE FOR ALL CIRCUITS  \n");

      f=open("/usr/control/loss2",2);         /*Load the standard message */
        printf("          Lossrate =");       /*loss rate.                */
        for(n=0;n<5&&(loss[n]=getchar())!='\n';n++);
        write(f,&loss[0],4);
      close(f);}
    printf("
          Changes to these values which are desired during the game
          should be made by re-entering 'Modify' or by editing file
          named 'lossrate' in the directory named 'control' using the
          password 'wargame'.


    printf("       TO CONTINUE, DEPRESS <cr>\n");
    a=getans(a);

four:
```

```
printf("%c0,clear);
printf("
        4.  Enter the rate at which you wish messages to be GARBLED
        during transmission.  Five equals five percent. Use integer
        values.

        TYPE '1' IF YOU WISH TO ENTER A SEPARATE  RATE FOR EACH
        CIRCUIT TYPE.  (1-5)

        TYPE '2' IF YOU WANT A STANDARD RATE FOR ALL CIRCUITS.

        Note:  If you do not wish to change values currently set,
               TYPE <cr>

f=open("/usr/control/garbrate",2);
for(n=0;n<2&&(getg[n]=getchar())!='0;n++);
if(getg[0]=='0){close(f);goto five;}
getg[n]=' ';
write(f,&getg[0],4);
close(f);

if(getg[0]=='1'){
  printf("     ENTER THE MESSAGE GARBLED RATES FOR EACH CIRCUIT 0);
  f=open("/usr/control/garb1",2);
  i=0;
  while(i<5){                              /* Load the message garbled   */
    i++;                                   /* rates for each circuit qual*/
    printf("0or circuit quality %d:0,i);
    printf("            Rate =");
    for(n=0;n<5&&(chgarb[n]=getchar())!='0;n++); chgarb[n]=' ';
    write(f,&chgarb[0],4);}
  close(f); }

if(getg[2]=='2'){
  printf("     ENTER THE STANDARD RATE FOR ALL CIRCUITS0);
```

```
      f=open("/usr/control/garb2",2);      /* Load the standard message  */
        printf("            Rate =");      /* garbled rate.              */
        for(n=0;n<5&&(garbage[n]=getchar())!='0;n++);
        write(f,&garbage[0],4);
      close(f); }

    printf("
            Changes to these values which are desired during the game
            should be made by re-entering 'Modify' or by editing file
            named 'garbrate' in the directory named 'control' using the
            password 'wargame'.

    printf("      TO CONTINUE, DEPRESS <cr>0);
    a=getans(a);

five:

    printf("%c0,clear);
    printf("
            5.  Enter the number of players and the name of each player
            who has been 'DESTROYED' or who for some other reason is to
            be removed from the game after the game has started.  Enter
            the NUMBER and a <cr> and the NAME of each player with a
            <cr> following each name.

            Note:  If you do not wish to change values currently set,
                   TYPE <cr>

    printf("Number of player(s) to be removed =");
    for(m=0;m<3&&(count[m]=getchar())!='0;m++); count[m]=' ';
    if(count[0]=='0){goto end;}
    g=atoi(count);
    f=open("/usr/control/destroyed",2);
    i=0;                                   /* List and load the names of */
    while(i<g){                            /* players to be removed from */
      i++;                                 /* the game.                  */
```

```
      printf("Player Name:    ");
      for(n=0;n<20&&(deadidx[i][n]=getchar())!='0';n++);
      deadidx[i][n]=' ';
      printf("      Removed  Player #%d is:  %s0,i,&deadidx[i][0]);
      seek(f,0,2);
      write(f,&deadidx[i][0],20);}
    close(f);
end:
    printf("RETURN TO THE MAIN MENU BY DEPRESSING <cr>0);
    a=getans(a);
    main();
}

/*  MESSAGE HANDLING SUBROUTINE  */

play(){
    int fk;
    int tvec[2],time(),*localtime(),*locl;
    char *ctime();
    time(tvec);

    printf("0s",ctime(tvec));
    printf("AFTER TYPING MESSAGE -- cntrl d0);
    printf("TO INCLUDE A FILE WITH YOUR TEXT -- cntrl b0);

    if((fk=fork())==0){
      execl("/bin/sndmsg","/bin/sndmsg",0);
      exit();}
      wait(&fk);
    main();
}
```

```
/*  MESSAGE HANDLING PROGRAM CALLED PLAY

        PLAY READS PARAMETERS IN FILES WHICH ARE ESTABLISHED BY
        THE PROGRAM WARGAME.  MESSAGES BUILD TO SEND ARE WRITTEN
        TO FILES WHICH ARE READ BY THE PROGRAM MAILER (PRIVATE VERSION).
        MAILER SENDS MESSAGES TO ADDRESSEES VIA THE PROGAM TRANS.
*/

char nameidx[20][20],deadidx[20][20],to[20],from[20],clear 014;
char sendchar[7],info[50];
int q,atoi(),matrix[20][20],rand(),qty,count;
float r,wq,atof(),sendtime;
double log();


main(){
  int f,i,j,k,n;
  char get[2],qual[2],names[20];


  f=open("/usr/cntrlf/number",0);
  read(f,get,4);                    /* Reads the number of players in   */
  close(f);                         /* the game.                        */
  qty=atoi(get);


  f=open("/usr/cntrlf/index",0);
    k=0;                            /* Reads the names of the players   */
    while(k<qty){                   /* in the game.                     */
      k++;
      read(f,names,20);
      for(n=0;n<20&&(nameidx[k][n]=names[n]);n++);
    }
  close(f);

  f=open("/usr/cntrlf/matrix",2);
```

```
  for(i=1;i<(qty+1);i++){
    for(j=1;j<(qty+1);j++){        /* Reads the comm quality matrix.   */
       if(i==j) break;
       read(f,qual,2);
       matrix[i][j]=atoi(qual); matrix[j][i]=atoi(qual);
     }
    }
  close(f);
  begin();
}

begin(){
  char a;

  printf("%c0,clear);

  printf("                ------   WELCOME TO WARGAME! ------0);
  printf("                    TO SEND A MESSAGE    TYPE 's'0);
  printf("                    TO READ YOUR MAIL    TYPE 'r'0);
  printf("                    TO EXIT THE PROGRAM  TYPE 'q'0);
  while(1){
  a=getans(a);
  switch(a){
    case's':case'S':
      message();
      break;
    case'r':case'R':
      mail();
      break;
    case'q':case'Q':
      stop();
      break;
    default:
      printf("TYPE 's' FOR SEND MESSAGE,     'r' FOR READ MAIL,
              'q' FOR QUIT!0);
      break;
```

```
 }
}

/*  MAILBOX READING SUBROUTINE  */

mail(){
  int fk;                                   /* Fork to 'msg' subsystem.*/
  printf("%c0,clear);
  printf("
         YOU WILL FIRST SEE A LISTING OF THE HEADERS OF MESSAGES IN
         IN YOUR MAILBOX.

              1.  TO SEE THE TEXT OF ANY GIVEN MESSAGE, TYPE  't'
                  FOLLOWED BY <esc> AND THE MESSAGE NUMBER. (Same
                  actions as in the 'MSG' system.)

              2.  AFTER READING MESSAGES TYPE 'q' TO RETURN.

              3.  DO NOT ATTEMPT TO SEND MESSAGES FROM THIS PART OF
                  THE GAME.

  if((fk=fork())==0){
    execl("/bin/msg","/bin/msg",0);
    exit();}
    wait(&fk);
  sleep(2);
  begin();
}

/*  MESSAGE HANDLING ROUTINE FOR INDIVIDUAL PLAYERS IN WARGAME  */


message(){
  int c,f,fk,g,i,j,k,n,r,s,t,cmpr,lost,garb,byte,que;
  char a,d,prec[20],lrate[5],val[5],line[7],words[20];
  char b[2],num[5],choice[2],grate[5],subj[50],text[2300],names[20];
```

```
char *txtptr;
int tvec[2],time(),*localtime(),*loc1,start[3],timenow;
int hrs,min;
char *ctime();
time(tvec);


f=open("/usr/control/destroyed",0);
  k=0;                                /* Reads the names of the players  */
  while(k<qty){                       /* who have been 'destroyed' or    */
    k++;                              /* otherwise removed from the game.*/
    read(f,words,20);
    for(n=0;n<20&&(deadidx[k][n]=words[n]);n++);
  }
close(f);
to[0]=from[0]=info[0]=' ';
c=n=m=s=t=byte=0;
printf("%c0",clear);


printf("                    MESSAGE HEADER:(no caps)    0);



printf("              %s0,ctime(tvec));
printf("0RCM?:");
for(c=0;c<26&&(from[c]=getchar())!='0;c++); from[c]=' ';

printf("00?:");
for(m=0;m<20&&(to[m]=getchar())!='0;m++); to[m]=' ';

for(s=1;s<(qty+1);s++){           /* Compares the addressee's name with   */
  n=0;cmpr=0;                     /* the list of 'destroyed' players to   */
  while(cmpr==0&&n<20){           /* prevent message transmission.        */
  cmpr=scmp(to[n],deadidx[s][n]);n++;
  }
```

```
  if(n==20){
  printf("COMMUNICATIONS WITH %s HAS BEEN LOST.0,&deadidx[s][0]);
  sleep(4); goto finish;}
}

printf("0NFO: (comma after EVERY entry)  ");
for(m=0;m<50&&(info[m]=getchar())!='0;m++);
info[m+1]='c';info[m+2]='o';info[m+3]='n';info[m+4]='t';
info[m+5]='r';info[m+6]='o';info[m+7]='l';info[m+8]=',';info[m+9]=' ';
for(k=0;k<c;k++){info[k+m+10]=from[k];} info[k+m+10]=' ';

printf("0UBJECT: ");
for(n=0;n<50&&(subj[n]=getchar())!='0;n++);subj[n]=' ';

for(s=1;s<(qty+1);s++){      /* Compares the sender's name with the  */
  n=0;cmpr=0;                /* list of players names to obtain a    */
  while(cmpr==0&&n<20){      /* value from the communications matrix */
  cmpr=scmp(from[n],nameidx[s][n]);n++;
  }
  if(n==20){break;}
}

for(t=1;t<(qty+1);t++){       /* Compares the address name with the  */
  n=0;cmpr=0;                 /* list of players names to obtain a   */
  while(cmpr==0&&n<20){       /* value from the communications matrix*/
  cmpr=scmp(to[n],nameidx[t][n]);n++;
  }
  if(n==20){break;}
}
q=matrix[s][t];               /* q is the circuit quality value (0-6)*/
printf("%d0,q);
if(q==1||q==2){               /* Circuits 1 and 2 have prec control. */
  printf("0RECEDENCE:");
  for(n=0;n<20&&(prec[n]=getchar())!='0;n++); prec[n]=' ';}
```

```
printf("
Message Text:  (Type '*'  When Finished)

          ----------------------------------
for(byte=0;byte<2300&&(text[byte] = getchar())!='*';byte++);
text[byte] = ' ';
printf("         0----------------------------------     0);

a=getans(a);                        /*  Dummy to erase the <cr>.        */
printf("Send or Abort?0);
a=getans(a);                        /*  Abort to the end of the program.*/
if((a=='a')||(a=='A')) goto finish;

if(q==0){                  /* If there is no direct link advise sender */
  printf("%c0,clear);
  printf("THERE IS NO DIRECT CIRCUIT TO    %s0,&tc[0]);
  printf("(Route the message via your organization.          0);
  sleep(4); goto finish;}
if(q==6){                           /* If there is no degrade, skip over*/
  wq=0; goto mid;}                  /* the queing calculations.         */

/* ---------- queing, lossrate and garbled rate calculations----------*/

f=open("/usr/control/msgrate",0);
read(f,choice,4);                   /*  Read message rate and make the  */
close(f);                           /*  queing calculations in the sub- */
if(choice[0]=='1') {calc1(q);}      /*  routines  calc1 or calc2.       */
if(choice[0]=='2') {calc2(q);}
printf("%3.3f0,wq);

j=k=garb=lost=que=0;                /*  Initialize appropriate values.  */
r=rand()/32767.0;                   /*  Read the rate for message losses*/

f=open("/usr/control/lossrate",0);
read(f,lrate,4);
close(f);
```

```
    if(lrate[0]=='1'){                  /*  Specific loss rates by circuit. */
      g=open("/usr/control/loss1",0);
      i=0;
      while(i<5){
        i++;
        read(g,val,4);
        if(i==q){j=atoi(val); break;}
      }
      close(g);
    }
    if(lrate[0]=='2'){                  /*  Standard rates for all circuits.*/
      g=open("/usr/control/loss2",0);
      read(g,val,4);
      j=atoi(val);
      close(g);
    }

    if(r<=j/100.) lost=1;

  r=rand()/32767.0;                     /*  Read the rate for garbled msgs. */

  f=open("/usr/control/garbrate",0);
  read(f,grate,4);
  close(f);
    if(grate[0]=='1'){                  /*  Specific rates for each circuit.*/
      g=open("/usr/control/garb1",0);
      i=0;
      while(i<5){
        i++;
        read(g,num,4);
        if(i==q) {k=atoi(num); break; }
      }
      close(g);
    }
    if(grate[0]=='2'){                  /*  Standard rates for all circuits.*/
```

```
        g=open("/usr/control/garb2",0);
        read(g,num,4);
        k=atoi(num);
        close(g);
      }
      if(r<=k/100.) garb=1;
/*-------------------------end queing calculations--------------------*/
mid:
  sendchar[0]=' ';                          /* Initialize sendchar.      */

  locl = localtime(tvec);                    /* After message is written,  */
  for(i=0;i<3;i++) start[i] = locl[i];       /* get the current clock time */
    hrs = start[2]*100;                      /* in hours and minutes.      */
    min = start[1];
  timenow = hrs + min;
  stime(hrs,min);                            /* Calculate the message send */
                                             /* time.  Sendtime equals the */
                                             /* current clock time plus the*/
                                             /* computed queing time (wq). */

              /* Write to files for reading by 'mailer' */

  if(lost == 1){                             /* If the message is lost:    */
    f=open("/usr/control/lostfile",2);
      seek(f,0,2);
      write(f,&sendchar[0],7);  write(f,"0,1);
      write(f,&to[0],20);       write(f,"0,1);
      write(f,&info[0],50);     write(f,"0,1);
      write(f,&subj[0],50);     write(f,"0,1);
      write(f,&text[0],2300);                /*  Write the message into a  */
      write(f,"0,1); write(f," ",1); /*  summary file.                    */
      close(f);
      goto end;
  }
  else if(garb == 1){  i=0;                  /* If the message is garbled: */
    while(i<byte){     i++;                  /* Randomly write a '&' over  */
```

```
    r=rand()/32767.0;                   /* the message text.          */
    k=0;
    if((r<=1.)&&(r>.8)) k=1;
    if((r<=.8)&&(r>.6)) k=2;
    if((r<=.6)&&(r>.4)) k=3;
    if((r<=.4)&&(r>.2)) k=4;
    if(r<=.2) k=5;
    text[i+k] = '&';
  }
  line[0]=' ';

  f=open("header1",0);
  read(f,line,7); close(f);             /* Read the first line of the */
  if(atoi(line)<=0) {                   /* header.  If it is empty,   */
    f=open("header1",2);
    write(f,&sendchar[0],7);
    write(f,&to[0],20);
    write(f,&info[0],50);
    write(f,&subj[0],50);
    close(f);
    g=open("text1",2);
    write(f,&text[0],kyte);             /*  write garbage into the    */
    close(g);                           /*  transmission file.        */
    goto end;
  }
  line[0] = ' ';

  f=open("header2",0);                  /*  If the first header file  */
  read(f,line,7); close(f);             /*  is full, check each other */
  if(atoi(line)<=0) {                   /*  header file until an      */
    f=open("header2",2);                /*  empty file is found.      */
    write(f,&sendchar[0],7);
    write(f,&to[0],20);
    write(f,&info[0],50);
    write(f,&subj[0],50);
    close(f);
```

```
  g=open("text2",2);
  write(g,&text[0],byte);
  close(g);
  goto end;
}
line[0] = ' ';

f=open("header3",0);
read(f,line,7); close(f);
if(atoi(line)<=0) {
  f=open("header3",2);
  write(f,&sendchar[0],7);
  write(f,&to[0],20);
  write(f,&info[0],50);
  write(f,&subj[0],50);
  close(f);
  g=open("text3",2);
  write(g,&text[0],byte);
  close(g);
  goto end;
}
line[0] = ' ';

f=open("header4",0);
read(f,line,7); close(f);
if(atoi(line)<=0) {
  f=open("header4",2);
  write(f,&sendchar[0],7);
  write(f,&to[0],20);
  write(f,&info[0],50);
  write(f,&subj[0],50);
  close(f);
  g=open("text4",2);
  write(g,&text[0],byte);
  close(g);
  goto end;
```

```
    }
    line[0] = ' ';

    f=open("header5",0);
    read(f,line,7); close(f);
    if(atoi(line)<=0) {
      f=open("header5",2);
      write(f,&sendchar[0],7);
      write(f,&to[0],20);
      write(f,&info[0],50);
      write(f,&subj[0],50);
      close(f);
      g=open("text5",2);
      write(g,&text[0],byte);
      close(g);
      goto end;
    }
    line[0] = ' ';
  }

  else{                              /* The message is not lost or garbled:*/
    line[0]=' ';
    f=open("header1",0);
    read(f,line,7); close(f);                /* Read the first line of the */
    if(atoi(line)<=0){                       /* of the header.  If it is   */
      f=open("header1",2);
      write(f,&sendchar[0],7);
      write(f,&to[0],20);
      write(f,&info[0],50);
      write(f,&subj[0],50);
      close(f);
      g=open("text1",2);                     /* empty, write the text to be*/
      write(g,&text[0],byte);                /* transmitted into it.       */
      close(g);
      goto end;
    }
```

```
line[0] = ' ';

f=open("header2",0);                    /* If the first header file is*/
read(f,line,7); close(f);               /* full,  check each other    */
if(atoi(line)<=0) {                     /* header file until an empty */
  f=open("header2",2);                  /* file is found.             */
  write(f,&sendchar[0],7);
  write(f,&to[0],20);
  write(f,&info[0],50);
  write(f,&subj[0],50);
  close(f);
  g=open("text2",2);
  write(g,&text[0],byte);
  close(g);
  goto end;
}
line[0] = ' ';

f=open("header3",0);
read(f,line,7); close(f);
if(atoi(line)<=0) {
  f=open("header3",2);
  write(f,&sendchar[0],7);
  write(f,&to[0],20);
  write(f,&info[0],50);
  write(f,&subj[0],50);
  close(f);
  g=open("text3",2);
  write(g,&text[0],byte);
  close(g);
  goto end;
}
line[0] = ' ';

f=open("header4",0);
read(f,line,7); close(f);
```

```
      if(atoi(line)<=0) {
        f=open("header4",2);
        write(f,&sendchar[0],7);
        write(f,&to[0],20);
        write(f,&info[0],50);
        write(f,&subj[0],50);
        close(f);
        g=open("text4",2);
        write(g,&text[0],byte);
        close(g);
        goto end;
      }
      line[0] = ' ';

      f=open("header5",0);
      read(f,line,7); close(f);
      if(atoi(line)<=0) {
        f=open("header5",2);
        write(f,&sendchar[0],7);
        write(f,&to[0],20);
        write(f,&info[0],50);
        write(f,&subj[0],50);
        close(f);
        g=open("text5",2);
        write(g,&text[0],byte);
        close(g);                          /* If all of the header files */
        goto end;                          /* are full, advise the player*/
      }                                    /* to resubmit his message.   */
      line[0] = ' ';
      printf("ALL THE COMMUNICATIONS CIRCUITS ARE BUSY.  PLEASE WAIT0);
      printf("A FEW MINUTES AND RESUBMIT YOUR MESSAGE.           0);
      que=1; goto skip;
    }
end:
      printf("Message to %s:  Sent0,&to[0]);
```

```
skip:
                    /*  Data collection segment.  */

  f=open("/usr/control/data",2);
  seek(f,0,2);
  write(f,&from[0],20);      write(f,"  ",2);
  write(f,&to[0],20);        write(f,"  ",2);
  write(f,&sendchar[0],7);   write(f,"  ",2);
  write(f,&subj[0],20);      write(f,"  ",2);
  anynum(wq);
  write(f,&sendchar[0],7);   write(f,"  ",2);
  if(lost==1) write(f,"LOST",4);
  if(garb==1) write(f,"GARB",4);
  if(que==1)  write(f,"WAIT",4);
  else        write(f,"SENT",4);
  write(f,"0,1); write(f,"  ",1);
  close(f);

  sendchar[0]=line[0]=' ';
finish:
  printf("TO CONTINUE, DEPRESS <cr>0);
  d=getens(d);
  main();
}


/* SUBROUTINE TO CALC THE TIME A MESSAGE SHOULD BE SENT TO ADDRESSEE */


stime(x,y)int x,y; {
  float addtime,temp;
  int i; char j;

  addtime=wq+y;                          /* y is the number of minutes */
```

```
    i=0;                                    /* past the hour.  If y + wq  */
    if(addtime>60){                         /* is greater than 60, add the*/
     i++;                                   /* appropriate number of hours*/
     addtime = addtime-60;                  /* to x to get the clocktime  */
     while(addtime>60){i++;addtime=addtime-60;}
     sendtime = (x+(i*100)) + addtime;      /* to send the msg (sendtime).*/
     }
    else{sendtime = x+y+wq;}
    anynum(sendtime);
  }

  /* CHANGES A NUMBER TO A CHARACTER STRING FOR VALUES <= 9999.99 */

  anynum(t) float t;{
    float temp; int i,j;
    temp=t;
    i=0;
    for(i=1;i<11;i++){
      if((temp - (i*1000))<=0){
        j=i-1 + '0';sendchar[0] = j;  break;}
    }
    temp = temp - ((i-1)*1000); i=0;
    for(i=1;i<11;i++){
      if((temp - (i*100))<=0){
        j=i-1 + '0';sendchar[1] = j;  break;}
    }
    temp = temp - ((i-1)*100); i=0;
    for(i=1;i<11;i++){
      if((temp - (i*10))<=0){
        j=i-1 + '0';sendchar[2] = j;  break;}
    }
    temp = temp - ((i-1)*10); i=0;
    for(i=1;i<11;i++){
      if((temp - i)<=0){
        j=i-1 + '0';sendchar[3] = j;  break;}
    }
```

```
    sendchar[4] = '.';
    temp = temp - (i-1); i=0;
    for(i=1;i<11;i++){
      if((temp - (i/10.0))<=0){
        j=i-1 + '0';sendchar[5] = j;  break;}
    }
    temp = temp - ((i-1)/10.0); i=0;
    for(i=1;i<11;i++){
      if((temp - (i/100.0))<=0){
        j=i-1 + '0';sendchar[6] = j;  break;}
    }
  }


  /*  CALCULATION SUBROUTINE FOR USER-DEFINED MESSAGE RATES  */

118 calc1(){
    int f;   char val[5], num[5]; float arr,ser;
    if(q==1){
      f=open("/usr/cntrlf/pick1",0);
      read(f,val,10);              /* Read average message arrival rate */
      close(f);
      arr=atof(val);
      f=open("/usr/cntrlf/pick1a",0);
      read(f,num,10);              /* Read average message service rate */
      close(f);
      ser=atof(num);
      distr(arr,ser,q);
      }

    if(q==2){
      f=open("/usr/cntrlf/pick2",0);
      read(f,val,10);
      close(f);
      arr=atof(val);
```

```
    f=open("/usr/cntrlf/pick2a",0);
    read(f,num,10);
    close(f);
    ser=atof(num);
    distr(arr,ser,q);
    }

  if(q==3){
    f=open("/usr/cntrlf/pick3",0);
    read(f,val,10);
    close(f);
    arr=atof(val);
    f=open("/usr/cntrlf/pick3a",0);
    read(f,num,10);
    close(f);
    ser=atof(num);
    distr(arr,ser,q);
    }

  if(q==4){
    f=open("/usr/cntrlf/pick4",0);
    read(f,val,10);
    close(f);
    arr=atof(val);
    f=open("/usr/cntrlf/pick4a",0);
    read(f,num,10);
    close(f);
    ser=atof(num);
    distr(arr,ser,q);
    }

  if(q==5){
    f=open("/usr/cntrlf/pick5",0);
    read(f,val,10);
    close(f);
    arr=atof(val);
```

```
        f=open("/usr/cntrlf/pick5a",0);
        read(f,num,10);
        close(f);
        ser=atof(num);
        distr(arr,ser,q);
        }
}

/*  CALCULATION SUBROUTINE FOR PRE-ESTABLISHED MESSAGE RATES  */

calc2(){
  float a,s; char val[8]; int f,n,cmpr;
  if(q==1){
    f=open("/usr/cntrlf/prerate",0);
    read(f,val,10);
    close(f);
    cmpr=scmp(val,"normal");
       if(val[0]=='n') a=1;
    cmpr=scmp(val,"medium");
       if(val[0]=='m') a=2;
    cmpr=scmp(val,"heavy");
       if(val[0]=='h') a=3;
    s=3.1;
    wq=a/(s*(s-a));
    }

  if(q==2){
    f=open("/usr/cntrlf/prerate",0);
    read(f,val,10);
    close(f);
    cmpr=scmp(val,"normal");
       if(val[0]=='n') a=1;
    cmpr=scmp(val,"medium");
       if(val[0]=='m') a=2;
    cmpr=scmp(val,"heavy");
       if(val[0]=='h') a=3;
```

```
    s=3.05;
    wq=a/(s*(s-a));
    }

  if(q==3){
    f=open("/usr/cntrlf/prerate",0);
    read(f,val,10);
    close(f);
    cmpr=scmp(val,"normal");
       if(val[0]=='n') a=1;
    cmpr=scmp(val,"medium");
       if(val[0]=='m') a=2;
    cmpr=scmp(val,"heavy");
       if(val[0]=='h') a=3;
    s=3.04;
    wq=a/(s*(s-a));
    }

  if(q==4){
    f=open("/usr/cntrlf/prerate",0);
    read(f,val,10);
    close(f);
    cmpr=scmp(val,"normal");
       if(val[0]=='n') a=1;
    cmpr=scmp(val,"medium");
       if(val[0]=='m') a=2;
    cmpr=scmp(val,"heavy");
       if(val[0]=='h') a=3;
    s=3.03;
    wq=a/(s*(s-a));
    }

  if(q==5){
    f=open("/usr/cntrlf/prerate",0);
    read(f,val,10);
    close(f);
```

```
      cmpr=scmp(val,"normal");
         if(val[0]=='n') a=1;
      cmpr=scmp(val,"medium");
         if(val[0]=='m') a=2;
      cmpr=scmp(val,"heavy");
         if(val[0]=='h') a=3;
      s=3.02;
      wq=a/(s*(s-a));
      }
}
/*  DISTRIBUTION TRANSFORM AND QUEING CALCULATION SUBROUTINE  */

distr(x,y,z) float x,y;int z;{
  float conv1,conv2,a,s;

  r=rand()/32767.0;
  if((r<.1)||(r>.9)) r=rand()/32767.0;
                                        /* Generate random number U(0,1).*/
  conv1= -(1/x)*(log(r));a=1/conv1;     /* Transform random number from  */
  conv2= -(1/y)*(log(r));s=1/conv2;     /* uniform distr to exponential. */
                                        /* Bound the distr by rejecting  */
  wq=a/(s*(s-a));                       /* uniform random numbers above  */
}                                       /* .9 and below .1  This bounding*/
                                        /* will eliminate excessively    */
                                        /* long or short queing times.   */


/*  CHARACTER FETCHING SUBROUTINE   */

getans(m){
  char ray[10];
  int i;
  for(i=0;(m=getchar())!='0;i=i+1) ray[i]=m;
  m=ray[0];
  return(m);
}
```

```
/*  PROGRAM EXIT SUBROUTINE  */

stop(){
  char b;
  printf("          CONFIRM YOU WISH TO EXIT THE PROGRAM. (Y/N)Ø);
  b=getans(b);
  if(b!='y') return;
  printf("                        THANK YOU AND GOODBYEØ);
  exit();
}
```

```
/*    ASYNCHRONOUS PROGRAM MAILER:

      -- COMPUTES THE CURRENT CLOCK TIME

      -- READS THE SENDTIME OF EACH HEADER FILE

      -- FORWARDS HEADER/TEXT INFO TO TRANS FOR
            MESSAGES WHOSE SENDTIME HAS EXPIRED

      -- ERASES HEADER/TEXT FILE AFTER FORWARDING

      -- OPERATES CONTINUOUSLY AND ASYNCHRONOUSLY
            WITH PLAY UNTIL PLAYER LOGS OUT
*/
int atoi();
float atof();
main(){
  int i,f,fk;
  char time1[7],time2[7],time3[7],time4[7],time5[7],to[20],info[50];
  char subj[50];
  int tvec[2],time(),*localtime(),*locl,start[3],timenow;
  time(tvec);


    locl=localtime(tvec);
    for(i=0;i<3;i++) start[i]=locl[i];
    timenow=(start[2]*100)+start[1];

    f=open("header1",2);
      read(f,time1,7);
      if((atof(time1) != 0)&&(timenow>=atof(time1))){
        read(f,to,20);
        read(f,info,50);
        read(f,subj,50);
        close(f);
        if((fk=fork())==0){
```

```
      execl("/usr/control/trans",to,info,subj,"text1", 0);
      exit();}
      wait(&fk);
      erase(1);
    }
    else close(f);

f=open("header2",2);
    read(f,time2,7);
    if((atof(time2) != 0)&&(timenow>=atof(time2))){
      read(f,to,20);
      read(f,info,50);
      read(f,subj,50);
      close(f);
      if((fk=fork())==0){
      execl("/usr/control/trans",to,info,subj,"text2", 0);
      exit();}
      wait(&fk);
      erase(2);
    }
    else close(f);

f=open("header3",2);
    read(f,time3,7);
    if((atof(time3) != 0)&&(timenow>=atof(time3))){
      read(f,to,20);
      read(f,info,50);
      read(f,subj,50);
      close(f);
      if((fk=fork())==0){
      execl("/usr/control/trans",to,info,subj,"text3", 0);
      exit();}
      wait(&fk);
      erase(3);
    }
    else close(f);
```

```
        f=open("header4",2);
          read(f,time4,7);
          if((atof(time4) != 0)&&(timenow>=atof(time4))){
            read(f,to,20);
            read(f,info,50);
            read(f,subj,50);
            close(f);
            if((fk=fork())==0){
            execl("/usr/control/trans",to,info,subj,"text4", 0);
            exit();}
            wait(&fk);
            erase(4);
          }
          else close(f);

        f=open("header5",2);
          read(f,time5,7);
          if((atof(time5) != 0)&&(timenow>=atof(time5))){
            read(f,to,20);
            read(f,info,50);
            read(f,subj,50);
            close(f);
            if((fk=fork())==0){
            execl("/usr/control/trans",to,info,subj,"text5", 0);
            exit();}
            wait(&fk);
            erase(5);
          }
          else close(f);

        dummy();
    }

    dummy(){
      sleep(30);
```

```
  main();
}

/*  SUBROUTINE TO ERASE A FILE AFTER IT HAS BEEN TRANSMITTED  */

erase(n) int n;{
  int f,g,i;

  if(n==1){
    creat("header1",0755);
    creat("text1",0755);
  }
  if(n==2){
    creat("header2",0755);
    creat("text2",0755);
  }
  if(n==3){
    creat("header3",0755);
    creat("text3",0755);
  }
  if(n==4){
    creat("header4",0755);
    creat("text4",0755);
  }
  if(n==5){
    creat("header5",0755);
    creat("text5",2755);
  }


}
```

LIST OF REFERENCES

1. Final Report, The OSD Colloquies on Military Command Control, p. i-ix, ESL, inc., 1979.

2. Stuart, Thomas E., LCDR, USN, "Command Control: The Two Edged Sword", U.S. Naval Institute Proceedings, p. 108-111, December 1980.

3. Miller, R.S., LTC, USA, Class Notes, U.S. Naval Postgraduate School, OS3670, 1980.

4. Theater-Level Gaming and Analysis Workshop for Force Planning, v.1, p. 129-134, Office of Naval Research, September 1977.

5. The Planning Process, 2nd ed., Annex I, Appendix 2, U.S. Naval War College, 1 August 1958.

6. Turban, E. and Meredith, J.R., Fundamentals of Management Science, p. 413-427, Business Publications Inc., 1977.

7. Andrus, A.F., Associate Professor, U.S. Naval Postgraduate School, Class Notes, OS 3655, 1980.

8. Holt, R.C., and others, Structured Concurrent Programming with Operating Systems Applications, Chapter 2, Addison-Wesley Publishing Co., 1978.

9. Trans.c a "C" language computer program, U.S. Naval Postgraduate School C3 Laboratory, 1981.

10. Kernighan, B.W., Programming in C-A Tutorial, Bell Laboratories, undated.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93940	2
3.	C3 Laboratory, Code 52 Naval Postgraduate School Monterey, California 93940	2
4.	Captain Wayne P. Hughes Code 55Hl Department of Operations Research Chair of Applied Systems Analysis Naval Postgraduate School Monterey, California 93940	1
5.	LtCol J.W. Johnson Code 39 Naval Postgraduate School Monterey, California 93940	1
6.	Professor John M. Wozencraft Code 74 Chairman, C3 Academic Group Naval Postgraduate School Monterey, California 93940	1
7.	Professor F. R. Richards, Code 55Rh Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
8.	Assoc Professor A.G. Andrus, Code 55As Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
9.	Assoc Professor N. Lyons, Code 55Ly Department of Operations Research Naval Postgraduate School Monterey, California 93940	1
10.	LT Ellen Roland, Code 55Ro Department of Operations Research Naval Postgraduate School Monterey, California 93940	1

11. Dr. M. Denicoff, Code 437 — 1
Office of Naval Research
800 N. Quincy Street
Arlington, Virginia 22217

12. LCDR A. J. Dietzler — 1
Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

13. LtCol L. Druffel, USAF — 1
Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

14. Dr. R. Kolb — 1
Naval Ocean Systems Center
Code 824
San Diego, California 92152

15. LtCol Thomas P. Stack, USAF — 1
SHAPE/SHOC
APO NY, 09055

16. LCDR Thomas A. Secorsky, USN — 1
Joint Strategic Connectivity Staff
Hq SAC
Offutt AFB, Nebraska 68133

17. Capt Dan A. Lynn, USA — 1
Combined Arms Combat Development Agency
ATZL-CAC-I
Fort Leavenworth, Kansas 66027

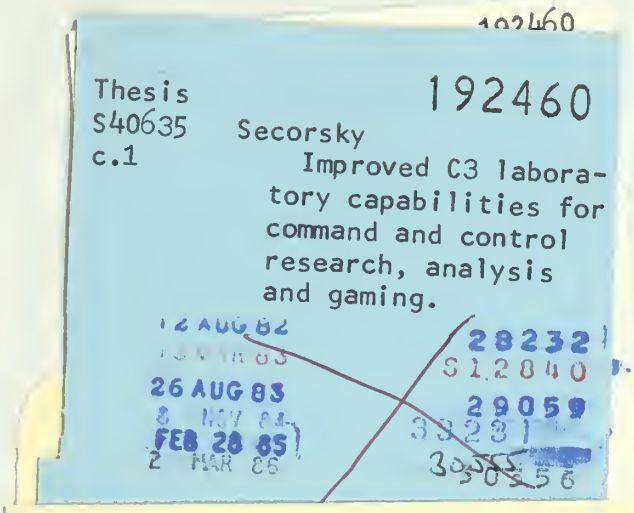
18. RADM L.S. Kollmorgen, USN — 1
Office of the CNO
OP-96
Washington, D.C. 20300

19. CAPT William Hunter, USN — 1
Office of the CNO
OP-942B
Washington, D.C. 20300

20. CDR R. Meinhold — 1
CINCPACFLT Staff
P.O. Box 6
Pearl Harbor, Hawaii 96860

21. Major Charles Earnhart 1
AFIT/CISP
Wright-Patterson AFB, Ohio 45433

22. Capt K. Moore, USAF, Code 62mg 1
Naval Postgraduate School
Monterey, California 93940



Thesis
S40635
c.1
192460
Secorsky
Improved C3 laboratory capabilities for command and control research, analysis and gaming.